HESEA-ACE
Reprint Articles from Other Newsletters

HESEA-ACE
Reprint Articles from Other Newsletters

Hardware Mod

SLCC 12/85

CLAUS BUCHHOLZ

A 130XE-COMPATIBLE 256K UPGRADE FOR THE ATARI 800XL

I designed the 256K upgrade described in my article, "The Quarter-Meg Atari" (BYTE, September, 1985), in December, 1984. Since this predated the 130XE, there was no precedent for extended memory on the XL's. I felt free to implement a system of eight 32K banks. The major reason was to keep the add-on circuit as simple as possible.

The 130XE, introduced in early 1985, set a different standard for bank-select memory. It uses 16K banks and makes them separately available to both the CPU and the video controller (ANTIC). The XE has 128K total memory. The 64K extended RAM is split into four 16K banks.

A 256K 800XL has 192K extended RAM, which requires 12 16K banks. I have designed a new upgrade for the 800XL that implements such a scheme. Its similarity to the 130XE's scheme allows use of software for the XE on a 256K 800XL.

To select one of four banks, the XE uses two bits, #2 and #3, in the memory control register (port B of the 6520 PIA, addressed at $D301 or 54017 decimal). Zeroing bit #4 makes the selected bank appear at addresses $4000-$7FFF (16384 to 32767 decimal), as seen by the CPU. Zeroing bit #5 makes it appear there as seen by ANTIC.

In my upgrade, bits #2, #3, #5 and #6 select one of the twelve banks. Zeroing bit #4 makes the selected bank appear at $4000-$7FFF to both the CPU and ANTIC. So, any program for the XE that uses the extended RAM for CPU storage will work on an 800XL with this mod. Those programs won't use the additional 128K, though. Programs that use the video banking feature of the XE might run on the modified XL, but the screen display will be wrong.

The procedure for this upgrade is basically the same as in the article, except for the following points. If your ANTIC (U7) part number is C021697, use the circuit of the figure, excluding the area inside the dotted lines. If it is the C012296, include the circuit inside the dotted lines. The circuit requires five connections to the PIA (U23). So pins 12 through 16 must be bent up and connected to the circuit. The rest of the procedure is the same. Notice that this circuit has one more chip than the article's circuit. This is the price of compatibility.

With the 256K dynamic RAMs in your XL, be sure to wait at least ten seconds after turning the computer off before you turn it back on. Otherwise it may not coldstart properly.

My original RAMdisk software doesn't work with this new mod. Enclosed is a listing of the new version. It is used in the same way, except that it offers a choice of either two single-density RAM disks or one double-density. If you wish a disk copy of the source and object code, send me a blank disk and return mailer with full postage, and I will promptly send it back with the software (Claus Buchholz, 201C East Edgewood, Lansing, MI 48910). Alternately, you may download the software from the Capitol Hill Atari Owners' Society BBS at 517-371-1106 or from the Castle Communications board at 517-371-4234. The source file is called QMEGXLD.SRC for Quarter-MEG XL Double.

Also available is a RAMdisk program that sets up one single-density RAMdisk and leaves the XE-equivalent banks free for XE software. This is quite useful with BASIC XE, DOS 2.5, or the new Synapse software. Its name is QMEGXLS.SRC.

I ask one thing in return for this information: please pass it around to all your interested friends. Put it in your club's library or on your favorite BBS. Encouraging software support of 256K will result in many interesting uses for it. Thank you and enjoy!

P.S. In response to an often asked question, I state that I have no documentation for my 192K upgrade for the 800. It involves modifying an Axlon 32K board to imitate a 128K Axlon RAMDISK and upgrading an Atari 16K board to 64K. It is a difficult mod, and I recommend the XL mod instead.

(SLCC JOURNAL EDITOR'S NOTES: We recommend reading the BYTE article mentioned above for a better description of the basic modification, then apply this article's information to your mod.)

PARTS LIST

8 41256 256K-bit dynamic RAM (200ns or less)
1 74LS153 Dual 4-to-1 multiplexer
1 74LS139 Dual 2-to-1 decoder
1 33 ohm, 1/4 watt resistor
1 Radio Shack # 276-150 circuit board
1 16-pin DIP header and short ribbon cable
3 16-pin low profile sockets

ADDITIONAL PARTS FOR ANTIC #C012296

1 74LS158 Quad inverting 2-to-1 multiplexer
1 74LS393 Dual 4-bit counter
1 16-pin low profile socket
1 14-pin low profile socket

DEFINITION OF MEMORY CONTROL REGISTER AT $D301 (54017 decimal)

XL MOD

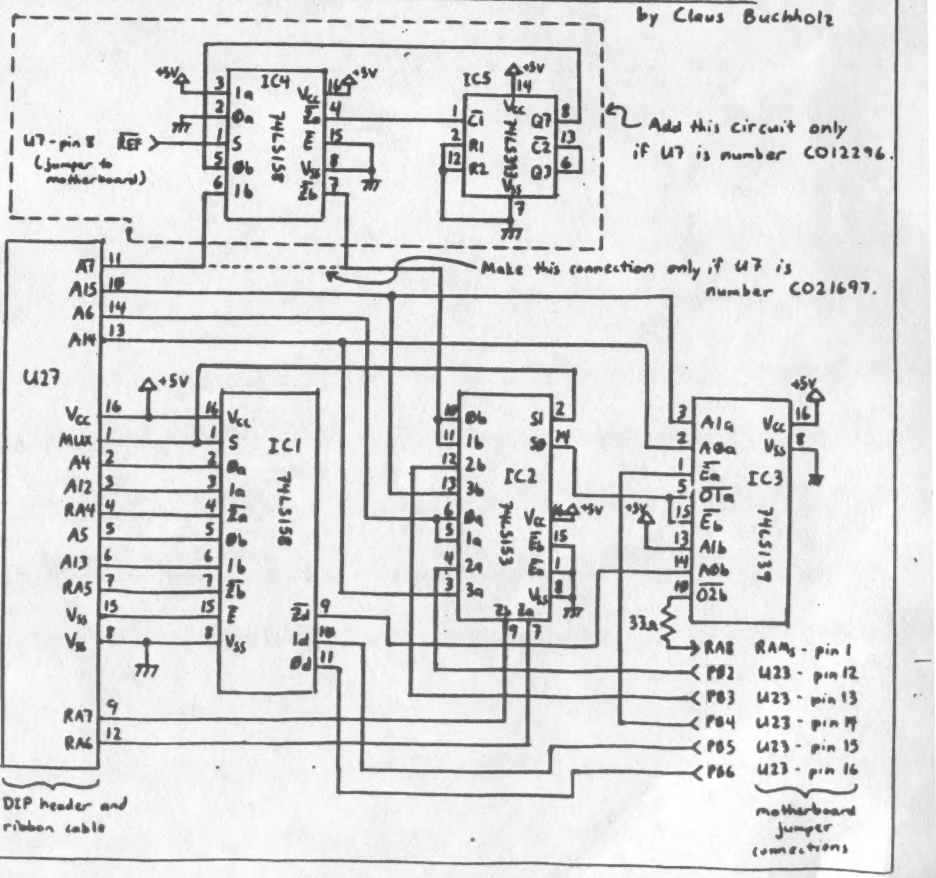
bit: 7 6 5 4 3 2 1 0
D a b E c d B R

D=0 enables diagnostic ROM
B=0 enables BASIC ROM
R=1 enables OS ROM
E=0 enables extended RAM
abcd is 4-bit extended RAM bank #
- ranges from 4 to 15
- banks 12 to15 are equivalent to XE's banks 0 to 3

130XE

bit: 7 6 5 4 3 2 1 0
D V C x y B R

D=0 enables diagnostic ROM
B=0 enables BASIC ROM
R=1 enables OS ROM
V=0 enables extended RAM for video
C=0 enables extended RAM for CPU
xy is 2-bit extended RAM bank #
- ranges from 0 to 3



(5)

ONE WOMANS' OPINION

By Diane M.Molnar

THE BUILDING BLOCKS OF ASCII

Computers, computers. What's becoming of this world? Everywhere you go there sits a computer!

I just started with a new firm as an Interior/Graphic designer and practically everything there is done on computers. So, what else is new? The one neat system they have can print out all the fonts of letters in various sizes and styles. We don't even have to go out and buy the letter types as often.

I had come across a very interesting point while using their computer one day. We forget that every letter we type into the computer transmits an electrical pulse. A pattern of an electrical pulse "on" is represented by a 1, and "off" by a 0. This system is known as ASCII (American Standard Code for Information Interchange).

When you hit the letter A on a typewriter keyboard, a hammer strikes the ribbon and makes the letter appear in ink on the page. The process is strictly mechanical. Hitting the same key on a computer keyboard, however, generates a set of zeros and ones, which causes the letter to appear as a luminous display on the screen. Every part of the process after the initial tap of the key is electronic. The zeros and ones used to encode the letter or any other character or control function are standardized. Computers can therefore pass information back and forth without translation. (They are using a shared electronic language (ASCII)).

There are a string of seven zeros and ones (binary digits, or bits)to every upper- and lower-case letter of the alphabet, to the numeral of the decimal system and to an assortment of punctuation marks and control symbols. An eight bit is either ignored or used as a check on the accuracy of transmission.

Seven bits provide 128 possible arrangements of zeros and ones. The first 32 are reserved for such codes as " carriage return" and "backspace" which are used to control screen displays and printers. The remaining 96 are called the printable codes because all but the first and last ,the ones for "space" and "delete", produce visible characters.

ASCII is constructed so that certain bits signal one piece of information ("this is a capital letter" or "this is a numeric character"), while the rest specify which letter and which numeral. The ASCII code for the capital letter A, for example, is decimal 65, which translate into binary 01000001. Lower-case is decimal 97, or 01100001; the difference is in the three leftmost bits. Below are examples of the ASCII binary code...try computerizing your own name.

```
A 01000001      j 01101010
B 01000010      k 01101011
C 01000011      l 01101100
D 01000100      m 01101101
E 01000101      n 01101110
F 01000110      o 01101111
G 01000111      p 01110000
H 01001000      q 01110001
I 01001001      r 01110010
J 01001010      s 01110011
K 01001011      t 01110100
L 01001100      u 01110101
M 01001101      v 01110110
N 01001110      w 01110111
O 01001111      x 01111000
P 01010000      y 01111001
Q 01010001      z 01111010
R 01010010
S 01010011      0 00110000
T 01010100      1 00110001
U 01010101      2 00110010
V 01010110      3 00110011
W 01011000      4 00110100
Y 01011001      5 00110101
Z 01011010      6 00110110
a 01100001      7 00110111
b 01100010      8 00111000
c 01100011      9 00111001
d 01100100
e 01100101    space 00100000
f 01100110      . 00101110
g 01100111      , 00101100
h 01101000      - 00101101
i 01101001      " 00100010
                ' 00100111
```

01010011 01100101 01100101 01111001 01101111
01110101 01100001 01110100 01110100 01101000
01100101 01101101 01100101 01100101 01110100
01101001 01101110 01100111 !!!!

P.S. Watch for November's article on Special Computer Printing.

(6)

Why Arrays? (Part 2)

by Richard Kushner

Nov 85 MCG

In the October issue of this newsletter we began our discussion on arrays. Please review to that article before continuing with the following discussion. We presented arguments why arrays are valuable, the rules for their use with Atari BASIC and explored one-dimensional arrays. Now we will go on to two dimensional arrays, expand on our examples and summarize.

So far we have used arrays as sort of one-dimensional lists. What if we wanted to not only include the uniform numbers of each Little Leaguer, but also their ages and telephone numbers? We could, of course, have three separate arrays to handle this. However, once again, Atari BASIC comes to our rescue with the *two-dimensional array*.

The best way to explain a two-dimensional array is to see one, as in TABLE 1.

TABLE 1

PLAYER	UNIFORM NO.	AGE	PHONE NO.
1	23	10	5551234
2	12	9	5559876
3	35	11	5554321
4	10	11	5556789

and so on for the other 21 players.

We have gone from a one dimensional list to a two-dimensional table. The "rules" we listed earlier, however, also apply to twodimensional arrays. We must DIMension the array with a statement like

```
10 DIM PLAYER(25,3)
or
10 DIM PLAYER(3,25)
```

Either way is correct, it is just a matter of how you prefer to visualize the array. In the first case, the array can best be thought of as a row across representing the 25 players, with the three pieces of information about each player listed underneath each player. In the second case, we have a column representing the 25 players, with three parallel columns containing the data of interest. The example above fits this second description. Thus PLAYER(3,4) in our example is the third piece of information (the phone number) of PLAYER(4), which is 5556789. Keep in mind that either layout can be used, but it will affect which elements store which information.

A word of caution. The mathematics of two-dimensional arrays gets somewhat abstract when you start manipulating items contained in the array. Your program may involve sorting the array and use a statement like

```
100 IF PLAYER(I+1,J+1)>PLAYER(I,J) THEN
PLAYER(I,J)=PLAYER(I+1,J+1)
```

You will be all right as long as you keep in mind the rectangular format of the array and which subscript refers to the rows and which refers to the columns. It is always good practice to first run your program with known data to be sure that the right numbers come out when known numbers go in. Arrays gone awry are a good example of the computer axiom "GIGO" - Garbage In, Garbage Out!

Let's further reinforce our growing knowledge of arrays with another example. As a present we received a weather station and we've been recording the temperature at 6:00AM, Noon and 6:00PM each day for one week. We want to write a program to accept all this information and then print it out in an orderly table, including the average temperature for each day. We know that the average temperature is just the sum of the three daily temperatures divided by three.

We have seven days worth of data and three measurements each day clearly a perfect candidate for a (7,3) array. LISTING 4 shows one way to write a program to accomplish our goals.

LISTING 4

```
90 REM * FIND AVERAGE TEMPERATURE
95 DIM TEMP(7,3)
100 FOR DAY=1 TO 7
110 FOR READING=1 TO 3
120 READ TEMP
125 TEMP(DAY,READING)=TEMP
130 NEXT READING
140 NEXT DAY
175 REM
180 PRINT "      TEMPERATURE"
190 PRINT "Day  6AM  12N  6PM  Avg."
200 FOR DAY=1 to 7
202 PRINT DAY;"   ";
205 TOTAL=0
210 FOR READING=1 TO 3
220 TOTAL=TOTAL+TEMP(DAY,READING)
230 PRINT TEMP(DAY,READING);"   ";
240 NEXT READING
250 PRINT TOTAL/3
260 NEXT DAY
980 REM
1000 DATA 76,79,75,72,77,76
1010 DATA 74,79,81,75,80,83
1020 DATA 80,77,70,68,65,65
1030 DATA 65,67,76
```

Lines 1000-1030 contain the temperature readings which are READ into the array using lines 100-140. We then print out the information in a table, using lines 205-250 to also calculate the average daily temperature. Note also the use of a REM statement in line 90 to indentify our program. Months from now, this will help us remember what the program does.

Observe the use of descriptive variable names (DAY, READING, TOTAL, TEMP) to aid in following the program logic. Little things like this mean a lot in program development and readability.

We could easily have expanded our array to include a temperature reading each hour, or included wind speed and relative humidity readings. Only our imagination (and our weather station) limits us! Figure 1 shows the results of running this program. We have been able to input the desired information into an array and output it in a concise form, including a calculation of the average temperature.

FIGURE 1

Temperature

Day	6AM	12N	6PM	Avg.
1	76	79	75	76.66666666
2	72	77	76	75
3	74	79	81	78
4	75	80	83	79.33333333
5	80	77	70	75.66666666
6	68	65	65	66
7	65	67	76	69.33333333

This ends our brief exploration of numeric arrays. You may have noticed that we have avoided including the names of the players in our Little League example or the names of the days of the week in our temperature example. This is because your Atari treats information that uses the letters of the alphabet (known as "strings") quite differently then plain, vanilla numbers. Atari BASIC does not have the ability to work with arrays of strings. We can use other properties of Atari BASIC to simulate string arrays, but that is beyond the scope of this article. Rather than go into that here, we'll close this discussion with a summary of what we have learned so far about numeric arrays.

An array enables us to manage a number of variables by using one variable name. Arrays may be one- or two-dimensional and are created with statements of the form DIM ARRAY(X) for one-dimensional arrays or DIM ARRAY(X,Y) for two--dimensional arrays. The array size is one larger than the number used because the computer starts counting with zero. Array elements can be used in BASIC statements wherever a simple numeric variable can be used. With arrays we will often find it convenient to use FOR....NEXT loops to process all elements or a block of the elements.

Now go to it! You will undoubtedly find many uses for arrays in your own programming. Keep the rules in mind and the power and utility of arrays will be yours to command.

This article is based on the book Basic Atari BASIC, by James S. Coan and Richard Kushner, published by the Hayden Book Company, Hasbrouck Heights, NJ and available at bookstores and computer stores nationwide. It would make a wonderful Christmas or birthday present. The article is reprinted from "The Atari Explorer " magazine.

PEEKS AND POKES

by Kenneth J. Pietrucha - JACG

By now most of you are familiar with the Basic sound command SO.C,T,D,V. C is the channel that you want to turn on and is a number between 0 and 3 (four channels). The T stands for the tone or note we want to hear and is a number between 0 and 255. D controls our distortion level and is an even number between 0 and 14 (only the numbers 10 and 14 give a pure tone). Volume is controlled by a number between 0 and 15, where 15 is full volume. If you type the statement SO.0,100,10,15 and hit return, you will hear a tone which will not turn off until you cancel it with the statement SO.0,0,0,0.

We can get the same results by poking certain locations. Once again we have four channels, only this time they are selected by Poking an even location between 53760 and 53766 with the tone number between 0 and 255. Note that 53760 corresponds to channel 0, while 53766 is for channel 3. A Poke 53760,100 causes the same note to sound from channel 0 as the Basic sound command.

Before we get sound we must set our distortion and volume level. This information is poked into the odd numbered locations between 53761 and 53767, with the lowest location for channel 0.

The number to Poke for distortion and volume in our example must be calculated by the following relationship (16*Distortion) + Volume. For a distortion of 10 (Pure Tone) and full volume of 15, the number to be poked in location 53761 is (16*10) + 15 or 175.

The basic command SO.0,100,10,15 can be duplicated with a POKE 53760,100:POKE 53761,175. To turn the sound off, set the volume back to 0 with a POKE 53761,160 with the 160 being equal to (16*10) + 0.

If you wanted the same results from channel 3 (really the fourth channel), you would Poke registers 53766 and 53767. It's really pretty simple and much easier to use than the Basic sound command.

If you enjoy good music, you will soon discover that some of the notes generated by the Atari are a few cents short of a dollar.

In a future column I'll show you how to combine two channels for double precision sound.

THE DISK LIBRARY

NEEDS YOUR CONTRIBUTION

Share Your Original Program With Us

MACE 12/85

MODEL 1702 COLOR MONITOR

by Commodore

A Review by Moe Demming

Your first reaction is probably one of shock. Imagine, a Commodore product reviewed in these hallowed Atari pages. But this monitor is the definative Atari monitor. Not that much of a contradiction, when you consider that Jack Tramiel now owns Atari, eh?

This monitor was one of the first for home computers that comes with both a standard video/audio input and an audio/chroma/luma input. For me, this is fantastic. I have used the video/audio input as a monitor for my video taping sessions, and use the audio/chroma/luma input for my Atari. A rear mounted switch allows for switching between the two.

The reason the 1702 is the definative monitor in my opinion is because of the chroma/luma option. The Atari splits off these signals for monitor use, so why not use them? If Atari had introduced a monitor in their early product lines, it would probably have had the chroma/luma option. As it is, I never question the colors portrayed on my monitor.

The controls are hidden by a door that lowers to reveal them. Only the signal select switch is in the rear. This switch controls which input you wish to use...video/audio is in front, the computer input is in the rear. Tint, color, bright, contrast, horizontal position, vertical hold and volume are all up front and easy to get at. Also the power button protrudes for easy access, and a power-on indicator is also up front. I like that because when the computer is off and the monitor is on, you don't see raster like on a television set. Just a dark screen. A handy reminder to power-down when finished.

My only gripe is that the speaker is on top and points up. Seeing I have my computer in the basement and do much of my computing at night when the kids are asleep, that just points the sound straight up. I keep a cover over the speaker and that cuts the sound a little. A side mounting of the speaker would have been better in my case.

This monitor used to be around $250 in price (like when I bought it). Now it can be found in the $175-$200 range. It is a worthwhile investment in a computer as good as the Atari. I even understand that Commodore's look good on it as well...

ARE YOU BINARY, HEX OR DECIMAL?

Bay Area A.U.G.

- Advertising Department -

Commodore Amiga vs Atari ST
First Impressions
MACE 12/85

This review is placed in public distribution 1 Sept, 1985 for the interest of its readers. It may not be reproduced or quoted in any manner without the express written permission of its author.

CIS 76703,2007
(416) 881-9941

The following is NOT a review of the Amiga A-1000. It is a first impression of this machine, after having spent a day or so with it. It is also a comparison with the Atari 520ST, its most logical competitor. So that noone feels left out I'll also throw in some gratuitous comments on the Mac by way of comparison, (how to make LOTS of enemies).

The full specifications for both machines have been published now and initial reviews have appeared in the magazines. It is not my intention to do a side by side technical comparison. Anyone who wishes to do this can do so on their own, based on the published specs.

I am writing this then from the point of view of a consumer who might be lucky enough to bring home one of each to examine and play with for a day. I've had an ST around for a couple of months and have become familiar with it, but to this date, just as everyone else, haven't a single piece of application software (other than our developmental prototypes). The Amiga that I received does not come with any applications software (though the production versions will) and thus I'm pretty much in the same boat with both machines; able to play with the desktops and examine the development environment but not do anythting productive or useful. I'm going to be getting samples of all the early software next week, but comparing them without application software seems an appropriate and fair thing to do at this early stage in the lives of both machines and in some ways helps prevent the clouding of features by a particularly strong piece of software on one or the other.

I also want to stress that my perspective on these computers is that of someone who would primarily use them in a personal productivity environment. These are not in my opinion "home" computers (whatever they are) and they are obvioulsy not destined for the desks of corporate users in the Fortune 1000 companies. These are machines for the likes of you and I to use to do real world tasks; writing, filing, calculating, communicating and recreation and that will be the perspective of this piece. If you are approaching these machines from any other perspective then be warned that we differ.

First some caveats and disclosures. Though I am V.P. of Product Development for Batteries Included (BI) the following observations and opinions are my own and not those of BI. BI has a vested interest in the success of both computer systems as we currently have products under development for both of them.

I've been a long time Atari enthusiast, but on the other hand, I have followed the Amiga for over two years, since it was first shown behind closed doors to a few industry insiders as a collection of circuit boards being run by a mini under the table. I've lusted after one since, and followed the product's development very closely.

So with that out of the way, here goes. Remember! This isn't a technical analysis, it's a user at home playing with the machines and trying to draw some comparisons and conclusions.

Keyboard first. I like the Amiga's keyboard a great deal. In fact I would rate it as the second nicest keyboard I've ever used. Since I'm not a touch typist the exact position of keys doesn't concern me. I use so many different ones that the odd displacement is less disconcerting than poor tactile feedback, which on the Amiga is very good. The ST by comparison is acceptable but not quite as crisp. As a fairly fast two finger typist I find both quite acceptable.

The RETURN key is large and well placed and the FUNCTION keys well separated (which is one thing I'll quibble about on the ST). The ST keyboard, while not having quite as good a feel, is a direct copy of the DEC VT200 keyboard in terms of layout and thus will

appeal to many. Of course the ST is a one piece unit; the keyboard and computer are one while the Amiga has a detachable keyboard that connects through a telephone cable with modular jacks. If you're the type of user that likes to type with the keyboard on your lap then the Amiga will be prefered. Both the ST and the Amiga blow the Mac away when it comes to keyboards, by the way. The Mac simply doesn't have enough keys and I find it's angle uncomfortable.

Drives second. Both machines use the new 3.5" microfloppies and I love them. If I never see another 5 1/4" diskette it won't be too soon! I can't wait for the PC2 which will have them as well so that the rest of the industry is forced to make the switch.

The Amiga's drives are very high density, double sided, 880K formatted. The ST's that are shipping now are single sided 360K formatted. Supposedly Atari is to ship their double sided drives shortly. They'll be 720K formatted. A little smaller in capacity than the Amiga's, but in the same ballpark. This is REAL storage capacity folks, on either machine.

One curious thing is that the Amiga's drives seem to be running all the time. They are "almost" silent but an occasional 'clunk' can be heard that indicates that they are spinning. This doesn't bother me one way or the other since many system's drives are always 'on'. I don't think any conclusions regarding longevity or reliability can be drawn at this point. The Mac uses 360K formatted disks as well, so it is comparable to the ST. Most PC and Mac owners are starting to bemoan their small disk capacity so I think ST owners with single sided drives will end up feeling the same way. Apple supposedly has double sided drives on the way this fall and it's also likely than when IBM embraces 3.5" disks they'll be 720K or better as well.

The Amiga can accept a second outboard drive as can the ST. The Amiga's though doesn't need a separate power supply as it gets its juice from the system unit. Any additional drives must have a separate power supply though. AmigaDOS can address up to four external drives while the ST can address two as well as a hard disk.

Cosmetics: The Amiga isn't pretty but neither is the ST. The Mindset was pretty, but look where it got them. All in all I wouldn't choose either machine based on their cosmetics, both are very acceptable for either home or office. The Amiga's footprint is a bit larger but the keyboard slides under the system unit for storage and the monitor can sit on top of it (to a maximum of 40 lbs worth). The Mac is positivly ugly (in my opinion) by comparison.

Mice: Both machines come with rodents of the two button variety. There isn't much to choose between them. I have found the ST mouse's buttons to not be quite as crisp as I would like but that may just be the couple that I've used. While the ST has a two button mouse, GEM doesn't require the use of two buttons. Intuition does. The left button selects while the right button displays menu bars. I'm not crazy about this; in fact I dislike it! You thus have to keep the right button depressed while clicking on the menu item from the drop down with the left button. It's not easy (although I suppose I'll get used to it) and I much prefer the ST-GEM method! Non-GEM ST software can and will use the second button though so it isn't redundant.

Surprise. The Amiga has a fan, (Steve Jobs would have hated it). I would have as well except that it's without a doubt the quietest fan I've ever heard (not heard?) on a computer. I have a PC clone on my desk at work which has a fan so loud as to sometimes interfere with conversations. Most PCs are similarly loud. The Amiga's fan is totally inaudable even in a quiet home so no one should be bothered by it. The ST on the other hand does not have a fan and thus is obviously quieter still. The ST's drives don't spin all the time either, so all in all for someone who is neurotic about noise, the ST is preferable. Whether the fan will contribute to the Amiga's longevity remains to be seen. The Mac doesn't have a fan either, but I am told that the next generation modular Mac will.

Sound: The Amiga's sound capabilities are superb. It also has voice synthesis built in which is very intelligable. The ST's sound capabilities are alright but not in the same league. For games and music software applications the Amiga will shine. When it comes to personal productivity applications music is almost irrelevant. The Amiga has

stereo audio output and users can anticipate some exciting software that utilizes this capability. The ST though has a direct MIDI connection built in. I personally wouldn't buy one or the other simply on the basis of sound so I'm probably not the best person to comment on this.

Graphics! The Amiga defines the state of the art in affordable graphics capabilities. I won't get into the details since they've been detailed in print elsewhere (Issue #1 of Amigaworld; Creative Computing and Byte September issues for the Amiga and recent Antic's and ANALOG's for the ST). Cost aside for the moment, if graphics are your "thing" then the Amiga is superior to every other computer currently on the market or yet announced.

Now that there are Amiga's on dealers shelves [there are? --ed.] there is discussion regarding the ST's versus the Amiga's text display on their respective RGB monitors. At the desktop (Workbench) level, both machines are in 640X200 4 color mode. There appears to be some significant difference between the displays with the Atari color display looking crisper and easier to read text.

It is too soon to declare definitively what causes this but it may be in part that the Amiga's screen is quite a bit larger than that of the Atari monitor and thus one is able to more clearly see the the black inter-scan line stripes which reduces apparent resolution. Also, the Atari's font seems somewhat more pleasing to the eye.

One clear advantage that the Atari ST has is in its high-res monochrome mode, 640X360. This mode is incredibly crisp and readable, ideal for serious word processing and other long session uses. A separate special Atari monochrome monitor is required to use this mode but it is every bit as good if not better than the IBM PC's monochrome text mode or that of the Macintosh. The Amiga doesn't have any similar mode and this is a definite drawback for serious text applications. Many people with IBM type color displays (the same resolution as the medium res Atari and the Amiga) find it difficult to work all day at that resolution and end up getting monochrome cards for text work.

Now for the counterpoint. I don't know about you but I'm not able to spend from $1,000 to $2,000 or more (depending on system and options) on a game machine. Sure I like to play games, but I don't think that's the raison d'etre for either machine. It would be facile and unfair though to simply latch onto the Amiga's superior animation-graphics and sound and dismiss the ST. The financial part of the equation is significant and will be disussed in detail at the end of this piece, but it is a major consideration.

The graphics on the ST are not shabby by any means (I know first hand because we have a graphics program for the ST under development that really makes it shine). The bottom line? The Amiga is the winner in terms of graphics and sound without regard to price. How important this is is up to each user to decide. The 520ST is the clear winner for displaying text but the user must buy a second monitor or forsake color. The Mac doesn't have color capability (yet) so isn't in the same league. Monochrome (hi-res) graphics on the ST are as good if not better resolution than the Mac as well.

User Interface! This is a tough one. I like GEM very much. It is very close to the Mac in style and manner of use. In fact, in some areas GEM has features preferable to those on the Mac; variable size scroll boxes for example and the upper right sizing button.

Intuition on the Amiga is quite similar in style to GEM and the Mac. It has windows with scroll bars, a close button and several other "gadgets" are available. In many ways though it is quite different. For example on the Mac and ST-GEM, if you have a number of windows on screen you simply click on any visible part of a window to both bring it to the front and make it the active window. On the Amiga clicking anywhere in a window makes it the active window but does not top it (bring it to the top of a multi-window display). To do that you need to click on one of the upper boxes in the upper right hand corner of each window that places a window in foreground or background. I can't say that I like this method.

Otherwise Intuition is very Mac-GEM like. Close and size boxes are where you'd expect them; windows are dragged in a similar

manner, there's even a Trash can and Preferences (control panel) window for mouse, screen and keyboard settings. Once you know how to use any one of these systems (Mac-GEM-Intuition) you'll be able to use any of the others without a hitch. Finally (!) there is now virtually a standard user interface for microcomputers.

One thing that I've noticed is that the Amiga does disk IO every time that you change something on the Workbench (Desktop). Thus it is more akin to the Mac than GEM in it needs to talk to the drive frequently. Disk IO speed though seems to be extremely fast, about the same as the ST though I've not run any speed comparisons. Subjectivly they seem to be about the same and both appear to be much faster than the Mac.

One area where the Amiga is different is that besides having windows it has Screens. Unlike a Window, a Screen must be the full width of the display. Screens allow the Amiga to display different tasks in different resolutions. You can thus have one part of the screen in low-res multi-color mode playing a game while the bottom half is in hi-res running a word processor. A very nice capability indeed which brings us to multitasking.

The Amiga is a true multitasking computer. That means that it can run several separate tasks or programs simultaneously. For example, you're on-line on CompuServe on a conference. This can be quite boring, waiting for the other folks to type their thoughts. So, open a second Window and call up a game thus allowing you to play Cosmic Froggy Space Zapper during the dull moments. Just got a bright idea for that report due tomorrow morning? Open a third window for your word processor and write your report with the game and telecommunication conference running simultaneously.

This isn't the same thing as a desk accessory or a program like Sidekick. ALL of the programs are actually running at the same time, not just standing by on-screen. For me, this multitasking capability is the most exciting aspect of the Amiga and the one that means the most to me. I can barely walk and chew gum at the same time but there are many instances where I want to be able to run a couple of programs simultanioulsy (reply to E-mail while doing a compile, that sort of thing).

What about multitasking on the ST? Right now the ST can't do it but there is no reason why it shouldn't be able to. Multitasking is a result of the operating system used in the Amiga, not the hardware. Though I have no information to this effect, I wouldn't be surprised if Digital Reseach is considering or even working on a Concurrent GEM. They are working on Concurrent DOS 286 and since GEM is a key product for them marrying the two can't be far from their minds, particularly considering the threat of Topview and Windows.

Also Metacomco, the folks in England responsible for the multitasking Amiga OS, are known for the ease with which they are able to port their products to other machines. While the ST then doesn't have multitasking today, there's no real reason why it shouldn't at some time in the future.

Is multitasking worth the money for you? Only you can decide. The need for it is very much determined by the type of work you do and your work habits. I happen to find it a very exciting and useful capability. The Mac by the way is not multitasking and I've heard no rumors about Apple having such a capability in the near term.

An operating system is more than icons and windows. On the Atari ST (at least the machines that are currently being shipped to users) there is no way for the user to directly address TOS. All DOS commands must go through the GEM visual interface. Developers have received a Command processor and thus can access TOS directly. On the Amiga you open what is called a CLI, or Command Line Interpreter which then allows you to directly talk to AmigaDOS. A brief look at the DOS commands show it to be extremely powerful. But one apparent drawback is that all utilities appear to be disk based rather than RAM based. Thus you must have a DOS disk present all all times.

TOS is also a very competent operating system; based on CP/M 68K. Both of these are large, dense and difficult to learn and use, so in many ways the visual interfaces of GEM

and Intuition are a godsend. I hope that Atari sees fit to include their command interpreter with the ST in future as many serious users will miss having it.

Cost: The final frontier. This is what separates dreams from ownership. The equation is complex because of the number of variables and what comes with what machine. A 520ST has 512K of RAM but loses half of it to TOS/GEM needing to be booted off disk. When Atari finally ships the OS ROMS this will change but today a 520ST is really only a 256K machine.

The Amiga is at base level a 256K machine but one can buy a 256K board that plugs into a slot in the front giving you a 512K machine. Like Atari with the ST, the Amiga isn't ready yet to have its operating system ROMed. Commodore's approach though is to include what they call a Writeable Control Store; a hidden internal 256K board containing RAM into which the DOS and Intuition load. The user thus doesn't lose any RAM. On the other hand, Commodore has said that they have no intention of providing ROMS when they finally come out and early Amiga owners apparently will have to boot the "Kickstart" disk forevermore. The pre-release Amiga with 512K, by the way, shows 374,944 bytes free. Where 125K bytes have gone isn't immediately clear.

How ever you slice it, list price to list price with comparable displays, drives and monitors a 512K Amiga A-1000 with one drive and color monitor is almost twice the price of a comparably equiped Atari 520ST. Reportedly the Amiga will come with more bundled software, but then Atari has promised other software will be bundled as well. History has shown that bundled software is seldom the best, though end users usually end up purchasing the better software from independant developers. Conclusion? The Amiga is more expensive than the ST. I'll leave it to others to determine by exactly how much more. This also doesn't figure in discounts which will vary widely.

What that brings us to is the ultimate question (after the meaning of life, of course), which is, should I buy an Amiga or an ST? I know you're going to say "cop-out", but there is no one simple answer. Like your father used to say, "it depends".

The question of software availability aside for the moment; if money is no object, you'd probably buy the Amiga. But, only if serious and extended text display wasn't something important to you. Even without money as a consideration, the text display on the ST with the hi-res monochrome display is so good that it's a hands down winner in my book.

For color-animation, graphics and sound the Amiga clearly wins its turn. Even the most ardent Atari enthusiast will have to agree that the Amiga's three co-processors make it the pre-eminant graphics machine. The tradeoffs for this are price and the lack of a hi-res text mode.

The two remaining questions are corporate survival and software availability. Without software in both quantity and quality, no computer is worth having. Right now I may be regarded as lucky to have access to these two exciting new computers but I can't write this report on either as I don't have a word processor; I can't calculate their potential sales as I don't have a spreadsheet and in fact can't do a single useful thing with either.

Not fair you say? The Amiga will ship later in September with some basic productivity software and the first releases for the ST are also due. OK, but until there is sufficient software neither the ST nor the Amiga are anything more than pretty chunks of plastic and silicon. It took the Mac almost a year before there was sufficient quality software to make it a viable productivity tool. If no one ever wrote another piece of Mac software again the Macintosh would continue to be a useful computer. It may take at least a year until the same can be said for the ST and the Amiga.

What about the survivability of both Atari and Commodore. CBM's ills are well known. The C64 market is flat and they need the C128 and Amiga to be strong successes. Sales of eight bit Atari's aren't anything to write home about either but Atari pulled in its spending horns a long time ago and is lean enough (so we're told) to survive this period.

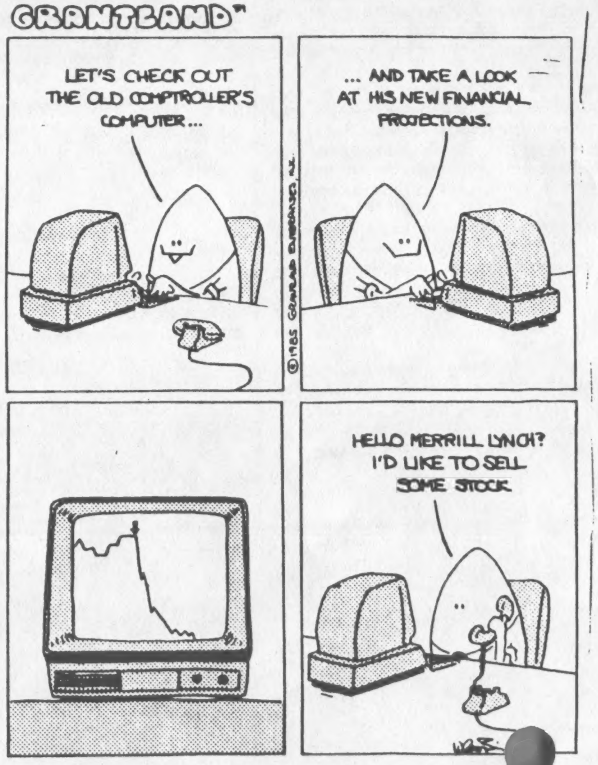
Whether the Amiga A-1000 and Atari 520ST sell enough to lift both the marketplace and

Commodore and Atari out of the doldrums still remains to be seen. Initial ST shipments appear to be selling well but clearly these sales must be to "early adopters" and closet software developers. Why would anyone buy such a machine with no software to run on it?

As this is being written in early September the Amiga has not yet shipped. Certainly when it does (supposedly later this month) many people will rush to buy it just as they did to snap up the first ST's. The real question becomes, after the initial "feeding frenzy" will there be sales to a broader base of more discriminating users? That remains to be seen, but the industry as a whole remains cautious. I for one am very confident that these two exciting micros will help to revitalize a lagging industry. Color me bullish.

weaknesses. If anyone tells you otherwise, he's lying. Clearly the A-1000 is not the ultimate Amiga nor is the 520ST the final ST. Both companies will be looking to push outwards in terms of both price and features; in both directions. Who's the benaficiary? You and I and all computer users.

If someone asks the question then, "which is better, the ST or the Amiga?", simply answer, "what's your budget and what do you want to do with it?" As for me, I'm waiting till they have models with 2 MEG of RAM in a lap-top design with color LCD display, 20 MEG 3.5" hard disk and all weighing less than 10 lbs. But on the other hand...



A sales rep from ATCOMP then demonstrated "HEX", a game similar to J-Sert on the 520ST. The meeting was rather noisy at that point so I didn't get any details. Sorry 'bout that.

The monthly door prize drawing featured a Novation SmartCat 1200-baud modem from Novation, printer paper and a printer head cleaner from Orange Micro, a package of disk drive head cleaners from COMPCO, Miner-49er from Big 5 Software, and a couple of joysticks from LAACE. Brad Pera was the lucky winner of the modem.

That's it for October. We hope to see you all at the November meeting but all turkeys should stay home!

***** FLASH *****

Are you an ST owner? Do you like the GEM Desktop the way it is? If so, that's too bad 'cause it's going to change and you can bet that Atari isn't very happy about it either. Apple has been threatening legal action over the similarity of the appearance and action of the GEM Desktop to the Macintosh Desktop. DRI apparently took that threat seriously and has agreed to make changes to GEM. DRI has provided a prototype of a new version of GEM to Apple but can not sell it until Apple determines that it doesn't violate any Macintosh patents. Apple may not care to make the evaluation a top-priority item. Meanwhile, Atari and other companies designing software products based on GEM software are in limbo and some feel that the futures of GEM and DRI themselves are doubtful in light of this development. It appears that Apple is insisting that the now-familiar ICONS such as the Trash Can, Disks, etc. must be changed. Even if Atari and DRI survive this latest threat, the GEM desktop will probably never be the same.

Disk Error Troubleshooting

by Bill Petry

LAACE NOV 85

Reprinted from R.A.G. BAG-Sept. 85

Many Disk related problems can be readily addressed by rechecking what you thought you checked or what you took for granted as being already done.

One of the worst culprits is leaving your diskettes in the drive when you power-up or turn the drive off. Always startup your drive before inserting your diskette. Always!! Always remove your diskettes before shutting down your system. It takes but one careless act to garble a sector header, checksum or worst of all the sector data itself. Checksums can be rewritten. Damaged sector headers are lost. Errored data are useless.

Another major source of grief is the much used write-protect bypass switch. Be very careful with these, especially when used with HAPPY or ARCHIVER enhancements. Sometimes the program (even though it is on the original disk) checks for normal load times, etc., and since the drive won't behave normally the program will think it's a copy and can really play all sorts of interesting havoc. Some programs have self-destruct subroutines built-in which will format the program disk when it senses anything out of the ordinary. All you'll hear is the familiar clunk, clunk, clunk ... but never quite soon enough to catch it in time.

If your trouble is with a copied disk you may have to make another copy from your original with a more sophisticated drive enhancement.

Tools of the Trade

Name	Author (Mfg)	Read Errored Sectors?
The Archiver	(Spartan Software, ICD)	Yes
Diskey	Sparky Starks (Adventure Int.)	Yes
Diskscan	David Young	No
Diskwiz II	Jerry Allen	Yes
Disk Doctor II	Steve Kaufman	No (Yes)
DiskTool	Tom Messina (Analog)	No
Disk Wizard II	(C.A.P. Software)	No
Scanalyzer	(Alpha Systems)	Yes
Sherlock	(4th Works)	No

The above listed disk utilities are rated only on their ability to read Cyclical Redundancy Checksums (CRC) and Bad Data Marked sectors. This ability is necessary to recover potentially useable data. My personal preference is Steve Kaufman's Disk Doctor II. It is written in BASIC and is listable so you can easily alter it to suit your needs. My copy is able to read errored sectors as well as print the disk directory.

Continued on page 16

Diskette Recovery

Before you get too excited or overwrought take the time to go over the following 5 startup reminders. When you are sure that these are all in order proceed to No. 1 and work your way thru the key to your solution.

STARTUP Reminders

?? Does program require BASIC?	Insert BASIC.
?? Does program require printer on?	Turn on printer.
?? OMNIMON present?	Turn OMNIMON off.
?? If ARCHIVER present is it on?	Turn off ARCHIVER.
?? Is drive a HAPPY drive?	Set to un-HAPPY.

Disk Error Solution Key

1a Message on screen 'ZAPPO' or 'BIG BROTHER IS WATCHING YOU' etc. Check STARTUP reminders then goto 14.
1b 'BOOT ERROR' on screen. Goto 10
1c Not 'BOOT ERROR'. Goto 2
1d Disk self-formats. Untrue copy or recheck STARTUP procedure.

2a Diskette has DOS files. Goto 3
2b No DOS files (boot disk). Goto 14

3a Error type 'SMARK' (more than two retries!). Goto 4
3b Error type 'BLIP-BLIP' (continuous). Goto 8
3c DOS Menu displayed? Goto 7

4a Directory readable with DOS. Goto 5c
4b Directory readable with DISK DOCTOR? Goto 5a
4c Directory hidden (DOS links present) Goto 5b
4d Directory unreadable. Goto 6

5a Directory sectors errored. Write them back with sector editor. Reboot.
5b Search for directory (Scanalyzer will do). Relocate dir. to fresh format disk. Goto 5c
5c Read directory with Disk Doctor and trace individual files- DOS.SYS, AUTORUN.SYS, (look at AUTORUN.SYS with sector editor to see next file that loads) etc. then if data present rewrite it back to sector. If DOS links messed up then goto 15.

6 DIRECTORY RECOVERY:
Copy existing track to different disk. Reformat track with ARCHIVER and write serviceable data back to it. Do directory recovery with DISK DOCTOR. Check programs for proper filenames and rename files for proper operation. Reboot.

7a AUTORUN.SYS present -insert BASIC. Reboot.
7b Trace AUTORUN.SYS. Goto 8

8 Trace disk to locate errored sector. Write sector data back to errored sector. Reboot.

10a DOS files. Goto 11
10b Not DOS files (boot disk). Goto 14

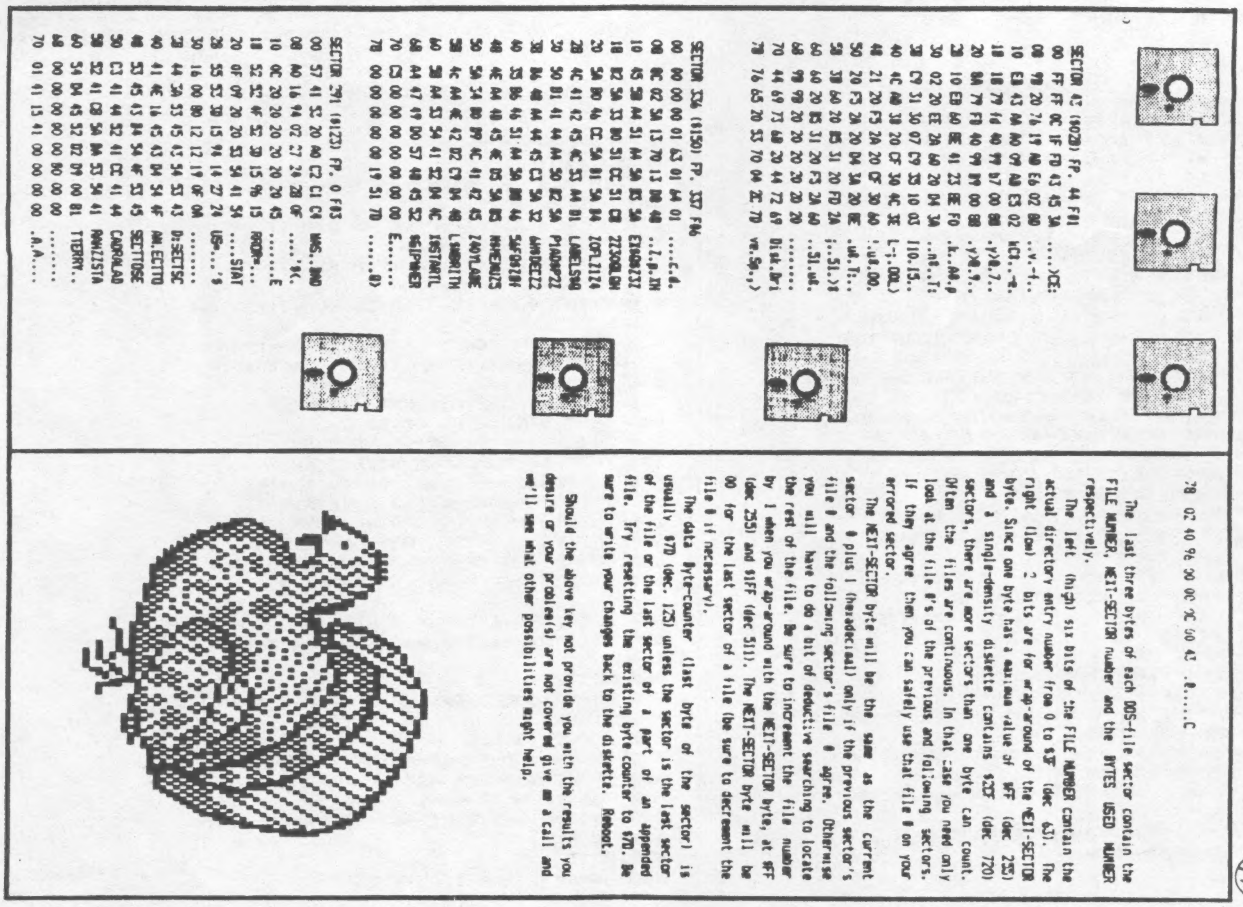
11a Sector read errored, but data present on 1st three sectors. Goto 8
11b First 3 sectors (boot record) unreadable. Goto 13

13 Copy 1st track (sectors 1 thru 18) of damaged disk to good disk. Reformat damaged track with ARCHIVER. Rewrite data back to original diskette. Copy the three boot sectors from freshly formatted diskette. If diskette contains unmodified DOS.SYS as first directory entry then entire first track may be written from good disk. Reboot.

14a Recopy disk from backup.
14b In some cases the entire track can be sector-copied to known good diskette and the original track reformatted with ARCHIVER. Write good data back to original. Writing $EA's (No Operation, NOP) to the entire cleaned sector sometimes will allow the program to run. If the data on the blank sector is part of a graphics display then it probably won't interfer with program operation. Reboot.

15 DOS LINKS:
When a DOS-file disk boots, the 3 'BOOT' sectors are read into the drive #1 buffer in the computer. These sectors contain the instructions concerning how the rest of the disk will be loaded and executed. If sector 4 contains the proper data (ie. it is the first sector of DOS.SYS), the computer will load it into memory. If not then the directory must be searched for DOS.SYS and when located it will be loaded and run. In DOS.SYS are additional commands for what to do next. First, the directory will be searched for AUTORUN.SYS, and if present it will be loaded and run. If no AUTORUN.SYS is found then the cartridge slot will be polled and if a cartridge is present, and it contains disk-booting instructions, the disk will then be booted, otherwise the cartridge will be initialized and run without further diskette access. If no cartridge is present, DOS will search the directory for DUP.SYS. This file will then be loaded and run. We now will see the DOS menu on the screen.

The three sector dumps that follow are sample displays from DOS files. The first is the beginning sector of a binary file showing the FF FF file header, the second is a starting sector from a BASIC tokenized file showing the BASIC file header, and finally an ending sector.



The last three bytes of each DOS-file sector contain the FILE NUMBER, NEXT-SECTOR number and the BYTES USED NUMBER respectively.

The left (high) six bits of the FILE NUMBER contain the actual directory entry number from 0 to 63 (dec 63). The right (low) 2 bits are for wrap-around of the NEXT-SECTOR byte. Since one byte has a maximum value of 255 (dec 255) and a single-density diskette contains 720 (dec 720) sectors, there are more sectors than one byte can count. Often the files are continuous. In that case you need only look at the file #'s of the previous and following sectors. If they agree then you can safely use that file # on your errored sector.

The NEXT-SECTOR byte will be the same as the current sector # plus 1 (hexadecimal) only if the previous sector's file # and the following sector's file # agree. Otherwise you will have to do a bit of deductive searching to locate the rest of the file. Be sure to increment the file number by 1 when you wrap-around with the NEXT-SECTOR byte, at 255 (dec 255) and 511 (dec 511). The NEXT-SECTOR byte will be 00 for the last sector of a file (be sure to decrement the file # if necessary).

The data Byte-counter (last byte of the sector) is usually 170 (dec. 125) unless the sector is the last sector of the file or the last sector of a part of an appended file. Try resetting the existing byte counter to 170. Be sure to write your changes back to the diskette. Reboot.

Should the above key not provide you with the results you desire or your problem(s) are not covered give me a call and we'll see what other possibilities might help.

DOCUMENT MAKER

12/85

by Ed Smith

Poor man's word processor Poor Man's word processor Poor man's word

This prograa is a poor man's word processor designed for writing documentation files. Line width is limited to 37 characters so that the file can output to the TV screen without double spacing. There is no right justification. Just type up to 37 characters at a time. If you need help, type H to get a Help Menu. It is a good idea to save DOC files with the suffix .DOC to all prograas. The Help menu shows up when you first boot-up and there-after only by typing H.

Help menu is as follows:

```
Correct a line
Delete a line
Help
Insert blank line
Save to disk
Top of file
```

Type only the first letter and RETURN to perfora the desired function. If you correct, delete or insert the entire file is listed with each line numbered. You aay preaaturely exit the listing by pressing the space bar. You then select the line nuaber. If you are correcting, then retype the line selected. Option T returns you to the top of the file so that you can view the file for errors and be sure the typing layout is proper and aeets your approval. Option S saves the documentation file to disk using the file name given.

It is suggested that upon completion of the documentation that the file be copied to the printer using DOS 2.5 Option C. It is good to have handy hardcopy for future reference.

Line Description

60	Set width of screen W and set dimensions and graphics.
70-90	Sense for *RETURN* key pressed
100-120	Print Help Menu
130	Error trap and ring buzzer.
140-190	Print numbered lines and get line # from user.

DOCUMENT MAKER Continued

200	Calculate X for right margin Print Help Menu and clear screen after sensing *RETURN* key pressed.
210	Get name of file. No "D:".
220	Use existing file option otherwise erase old file.
230	User has second chance to avoid erasing old file.
240	Print documentation
250-270	Get a line of documentation
280	Make this line blank.
290	Eliminate ? which occurs usually when back-spacing.
300-350	Six Help Menu Options
360	Make sure line is 37 bytes long by adding spaces.
370	Update memory and go back and get another line of documentation.
380-410	Disk file write routine.
420-430	Add spaces to make length of line exactly 37 bytes.
440-470	Disk file read routine.
480-520	Correction routine
530-570	Insert blank line routine.
580-640	Deletion routine.

Program Listing

```
30 REM PROGRAM DOCUMENTATION
40 REM BY ED SMITH NOV. 23,1985
50 REM DISK FILE D:DOCWRITE
60 W=37:DIM A$(0.8*FRE(0)),F$(16),P$(W
):GRAPHICS 0:SETCOLOR 2,11,1:GOTO 200
70 ? :? :? "HIT RETURN TO CONTINUE"
80 IF PEEK(764)=255 THEN 80
90 POKE 764,255:RETURN
100 ? :? "HELP MENU":? :? :?
```

Continued page 15

DOCUMENT MAKER continued

```
110 ? "Correct a line":? "Delete a lin
e":? "Help":? "Insert blank line":? "S
ave to disk":? "Top of file":?
120 RETURN
130 TRAP 40000:? CHR$(253);"OUCH!":GOT
O 100
140 FOR I=1 TO LEN(A$)/W:S=W*I-W+1
150 ? I;"  ";A$(S,S+W-7)
160 IF PEEK(764)=255 THEN NEXT I
170 POKE 764,255:RM=LEN(A$)/W
180 TRAP 130:? "ENTER LINE #";:INPUT P
$:LN=VAL(P$):IF LN<1 OR LN>RM THEN 130
190 RETURN
200 X=(W+1)*(W<39)+39*(W>38):POKE 83,X
:GOSUB 100:GOSUB 70:? CHR$(125)
210 ? "Name of file ";:INPUT F$:A$="D:
":A$(LEN(A$)+1)=F$:F$=A$
220 A$="":? "Use existing file (Y/N)";
:INPUT P$:IF P$(1,1)="Y" OR P$(1,1)="y
" THEN GOSUB 440:GOTO 240
230 ? "Are you sure ?";:INPUT P$:IF P$
(1,1)<>"Y" THEN IF P$(1,1)<>"y" THEN 2
10
240 ? A$
250 ? "Type your documentation"
260 ? "Enter up to ";W;" characters pe
r line.":?
270 ? :INPUT P$
280 IF LEN(P$)=0 THEN 360
290 IF P$(1,1)="?" THEN P$=P$(2)
300 IF P$="C" OR P$="c" THEN GOSUB 140
:GOSUB 400:GOTO 270
310 IF P$="D" OR P$="d" THEN GOSUB 140
:GOSUB 500:GOTO 270
320 IF P$="H" OR P$="h" THEN GOSUB 100
:GOTO 250
330 IF P$="I" OR P$="i" THEN GOSUB 140
:GOSUB 530:GOTO 270
```


DOCUMENT MAKER continued

```
340 IF P$="S" OR P$="s" THEN GOSUB 380
:GOTO 240
350 IF P$="T" OR P$="t" THEN 240
360 GOSUB 420
370 A$(LEN(A$)+1)=P$:GOTO 270
380 OPEN #1,8,0,F$
390 RM=INT(LEN(A$)/W):FOR R=1 TO RM
400 P$=A$(W*R-W+1,W*R)
410 PRINT #1;P$:NEXT R:CLOSE #1:RETURN

420 IF LEN(P$)<W THEN FOR I=LEN(P$)+1
TO W:P$(LEN(P$)+1)=" ":NEXT I:RETURN
430 RETURN
440 A$="":CLOSE #1:OPEN #1,4,0,F$
450 TRAP 470:INPUT #1,P$
460 A$(LEN(A$)+1)=P$:GOTO 450
470 TRAP 40000:CLOSE #1:RETURN
480 ? "Please retype the following"
490 ? :? :S=LN*W-W+1:? A$(S,S+W-1)
500 INPUT P$:IF LEN(P$)=0 THEN 520
510 IF P$(1,1)="?" THEN P$=P$(2)
520 GOSUB 420:A$(S,S+36)=P$:RETURN
530 RM=LEN(A$)/W:A$(LEN(A$)+1)="
                                "
540 FOR I=RM+1 TO LN+1 STEP -1
550 A$(I*W-W+1,I*W)=A$((I-1)*W-W+1,(I-
1)*W)
560 NEXT I
570 P$="":GOSUB 420:A$(LN*W-W+1,LN*W)=
P$:RETURN
580 RM=LEN(A$)/W
590 IF LN=1 THEN A$=A$(W+1):RETURN
600 IF LN=RM THEN 640
610 FOR I=LN TO RM-1
620 A$(I*W-W+1,I*W)=A$((I+1)*W-W+1,(I+
1)*W)
630 NEXT I
640 A$=A$(1,(RM-1)*W):RETURN
```


EDUCATIONAL COMPUTER USE PART 4

BY BOB MOSS

Continuing our discussion of educational programs and their use, let us consider simulations, and how to evaluate them. First, what type of simulation do you wish to offer the child? There are at least 4 broad types - historical recreations, modifications of historical events, fictions and fantasies with their own internally consistent rules, and representations of physical or scientific processes. Each type can teach something valuable, but not all programs are equally successful.

Historical programs often recreate major battles or explorations, such as Eastern Front, Legionaire, Battle for Normandy, or Seven Cities of Gold. If the simulation is properly constructed, all factors which affect the final result are considered, they are properly weighted, and if you follow the historical flow of events accurately, you will obtain the actual results. This means not only that you win or lose, but that lesson of war and materials clearly appreciate those of the actual forces involved. Also, the rules should be clear and the causes and effects of actions either clearly explained, or readily apparent during play. In order to really evaluate this type of program, you must either find a reviewer whom you can trust, or you must play the game with enough background on the actual event so that you can determine whether or not all the factors have been taken into proper account, and if the results parallel the historical event. Most people, and especially children, will not have an adequate historical background, and must trust the programmers to have done adequate research on the subject. In some cases, such as Eastern Front, all of the important factors appear to have been taken into account and given realistic weighting factors. There may not be proper consideration for mobility and surprise, but these are hard to quantify anyway. In general, if a historical simulation is true to the event it portrays, and clearly demonstrates the results of various logical actions, it is successful. Whether or not the child enjoys it is another matter.

Another historical simulation which is interesting is Seven Cities of Gold. The idea is to discover the Americas, colonize and conquer it, and return with as much of the wealth of the land as possible to Spain. If the child is familiar with the Spanish explorations of Central and South America, this is a very good program for playing "what if" games, and for directly comparing the results of different approaches - is it better to trade with the natives, bribe them, or fight them? What mix of supplies and men should be taken from Spain, and left as colonizers? Is it better to return again and again to the same places, or to constantly search for new places? If the game is played properly, the results of different actions can be seen as the game is played and saved repeatedly. The biggest problem with this program is that too little is known about the real cause and effect relationships during the Spanish conquests. Also the relations between tribes, local customs, effects of diseases, etc., all are neglected even though they had a major impact on actual historical events. Also, the manual gives almost no information on the actual events being simulated. It does refer the child to several reference works, but otherwise it provides little direct input on actual history. The best way to educate with any of these historical simulations is to have the child read a good survey book or article about the event before playing with it. Most of these simulation programs are for children from 9 to 12 years old.

CONTINUED ON PAGE 3

CONTINUED FROM PAGE 2

Omitted historical events take a real punch out on the basis for a program which either allows more choices and freedoms than the actual event, or which introduces a new factor(s) into the situation. While it is valuable to know and understand the actual event, in order to appreciate whether or not the simulation matches history, it is not necessary. Decisions made for the simulation may have entirely different inputs when compared to the real world. Therefore, it is important that parents know and understand what the program is intended to achieve. For example, Fortune represents the environment in which 19th century railroads were constructed, but it does not represent any particular railroad. The methods which the players use to accumulate the most wealth may be entirely different than those actually used by tycoons in those days. Rails West tries to more faithfully represent what happened to particular railroads. This type of program can be considered successful if the child gets a grasp on how business or exploration really work, and what people do in particular situations. Again, it is difficult to predict whether or not your child will co-operate and take realistic approaches towards solving the problems or still gloss over the program with little real interest in knowledge. It is important to work with the child on these types of programs, at least initially, and then have the child discuss how the program works, and suggest ways to improve it. Unless you have a clear goal when using this type of program, it is difficult to determine whether and how much the child really learned. In general, these types of simulations are most valuable as teaching aides, and to encourage general problem solving logic.

We will discuss the other 2 types of simulations next month, and suggest some programs which might be interesting and useful as teaching aides.

Forth In The Laboratory

by Donald Forbes - JACG

In today's undeclared submarine war under the oceans (Where do you hide or seek a missile bearing sub in an ocean canyon?), what would be your choice of a disposable computer to run sea-bottom instrumentation?

JACG member Dave Green thought you might be interested in the following highlights from an article on 'FORTH: The optimum language for microcomputers' by Professor Ferren MacIntyre of the Rhode Island University graduate school for oceanography (American Laboratory, Feb-Mar 85). He discusses the software and hardware, backed by an NSF grant, for an ocean-going bubble spectrometer to count and size naturally occurring bubbles in the surface ocean. Rockwell International has packaged a 6502 microprocessor, a Forth kernel, and much additional circuitry in the 40-point R65F11 chip with 16 I/O lines, which forms the heart of the disposable computer to run sea-bottom instrumentation. Here is Dr. MacIntyre's story:

The programming language Forth has, in the author's opinion, been overlooked as a useful laboratory tool. The purpose of this two-part paper is to acquaint readers with applications of Forth, and to suggest that time invested in learning Forth is well spent. Whether one is interested in saving space, minimizing compilation time, or rapid execution, Forth performs notably better than other languages in the microcomputer environment. Most projects spend more time in development than in execution, making fast compilation particularly valuable. The results of a benchmark run by a graduate student innocent of prior knowledge of computers, who is a fair surrogate for the average experimenter, showed that an IBM PC running a bubble sort program that took 2.1 Kbytes and 291 seconds in FORTRAN and 1.1 Kbytes and 36 seconds in Pascal took 700 bytes and 22 seconds in MMB Forth.

Originally, Forth was a FORTRAN program running on an IBM 1130, a 'third generation' computer of the late 1960s. The name was intended to convey the idea of a 'fourth-generation' language, but the 1130 would accept only five-letter names. The program was Charles Moore's response to his employers' penchant for changing mainframes periodically, and his intent was three-fold: to create a way of writing portable applications adaptable to any mainframe with a short machine-dependent kernel; to keep the language simple; and to make it easy to add whatever capability later became desirable.

That Forth turned out to be the ideal language for microcomputers may not be serendipitous, but rather a remarkable case of convergent evolution: good programming evolving independently in the same direction as good machine design. The next step in this evolution is a silicon chip designed to run Forth. Now in the prototype stage, this promises a 20-MHz clock and one Forth primitive per clock cycle -- speeds approaching the CRAY-1 supercomputer and 100 times faster than today's microcomputers.

Forth neatly links hand calculators and supercomputers. I have borrowed programs from both the HP-65 and the CRAY-1, translated them into Forth, and run them on an IBM PC, not as a stunt, but (in what is beginning to seem a characteristic of Forth programming) because this was the most efficient way to complete the desired job.

Have you ever wished that your calculator were not I/O limited? The statistical programs of the HP-65 (Hewlett-Packard) are so useful that I have written them in Forth (the language structure is very similar). The commands and documentation remain intact, but it is possible now to enter data from keyboard, memory, or RS232 line, obtain results 1000 times faster, display them on the screen or printer, save them to disk, or use them in further programs.

Again, have you ever tried accessing a supercomputer from a remote site? The 1000-fold speed advantage of the CRAY over the PC (with the 8087 numerical coprocessor) vanishes rapidly! An IBM task which takes all day in BASIC or 15 minutes in Forth will run in 1 second on the CRAY -- but it typically takes five minutes to get a usable telephone link to the CRAY and tell it what is wanted, reducing its speed advantage to 3:1. In any case, most CRAY jobs are run as background, with 24-hour turnaround, so for results needed today, I use the IBM and Forth.

Structure of Forth

Forth is not so much a language as a programming environment, or set of tools for generating good code. It is usually its own operating system (i.e., it initializes the machines, runs the I/O, formats and copies disks, has a debugger, assembler, editor, etc.). (Versions which run under MSDOS or CP/M may gain in file-handling but also inherit the inadequacies of the system, a trade-off necessitating careful thought.)

Forth uses stack-oriented, reverse-Polish notation (RPN), for the same reason that Hewlett-Packard chose it for calculators: RPN is one of the few ways in which people and computers can gracefully think alike. These structures are familiar from childhood, for RPN is the way we first learned to deal with numerical operations. RPN is much like introductory arithmetic: put a number on the blackboard (or stack), put another number below it, then write the operator, then replace the stack entries with the result. If it now seems strange, try to recapture the first time you saw an algebraic equation, and the feeling of helplessness that it induced! Like most forgotten skills, RPN can be relearned rather easily. The stack is simply a place to store things temporarily, organized exactly like its prototype on a first-in, first-out basis. Access to the top is easy; elsewhere, possible but less easy. It obviates the need for 'local variables,' and is the customary way of passing parameters.

Programming is a way of converting a sequence of logical ideas into a list of machine-executable instructions, or program. The device that does this is a 'compiler' if it produces permanent code, or an 'interpreter' if it is interactive. Forth has both: that is, if the user types 2 3 + . the interpreter will add the numbers with + ('plus') and display a 5 on the screen with . ('dot'). If the user instead types : TEST 2 3 + ; then : ('colon') turns on the compiler, which adds TEST to the dictionary, and ; ('semicolon') stops compilation. If we now type TEST the interpreter will find TEST in the dictionary and execute it, putting a 5 on the screen.

Traditional compilers struggle with the difference between human and computer thinking; BASIC minimizes the struggle by limiting its vocabulary; and Forth simply takes advantage of the similarities. The results are so striking that it is worth watching how this is done.

The classical way to associate human and machine thinking is through a dictionary, often as a 'vectored' list, in which the human names, or 'headers,' have pointers to their machine-code 'bodies.' A slightly more efficient storing is the 'in-line' dictionary, requiring the same space, but only one growth area. Pointers now link to the preceding entry. Another name for this is 'direct threaded code.' Note that like the original list of machine instructions, there are loops, branch points, and subroutine calls, because a string of instructions without decision structures would be so inflexible that it could not, for example, respond to a keyboard.

The final step is to reduce the machine code to a list of addresses rather than instructions. What this loses in scurrying, it regains in brevity, since most 'subroutines' will be reused. This 'indirect threaded code' is the structure of Forth.

We can now see that the power of Forth lies in its ability to convert ideas directly into programs, maintaining the original structure. Each of the subtasks -- Forth words -- like ACCEPT-INPUT is now decomposed into a set of simpler instructions, until we come to tasks so simple that the machine already knows how to do them.

Because it works with instead of against the programmer, the Forth compiler takes about four lines of Forth code. It and a small dictionary can be put into 1/K or Read Only Memory (ROM), permitting single chip devices to speak a high-level language.

'Writing a program' means adding new words to the dictionary, so that the new 'program' is in no way separated from what has gone before. Everything that the language can do, can be done from the middle of the program: Users may rewrite one sector of a disk, drive I/O ports, read an RS232 line, call upon some feature of the screen editor, dump memory, redefine a printer font, switch monitors, get a screen print, drop into assembler, inquire about plotter status, or perform any other function, provided only that its determining word is in the dictionary.

Lest this total control overwhelm available memory, FORGET <name> removes all words down through <name>. Other words can then be read from disk, providing unlimited virtual memory.

Forth instructions and data words share a common structure. Forth words can be lists of machine instructions, headerless code as in a vectored dictionary, or any number of varieties of data or other structures. In addition, there are defining words, or 'parents,' which create families of 'children' with various names but common behavior. These usually define new data types, so that if a user needs 7-dimensional arrays of pairs of 64-bit-wide complex numbers, there is a way to obtain them. Defining words are more complicated than ordinary words, because they must specify both the compilation-time behavior of the parent when it is creating the child, and the run-time behavior of the child when it is being used.

Pros and cons

Forth is not universally beloved, and some common criticisms include the following: Forth is a 'write-only language,' or a 'language for geniuses,' with a 'small user base' of 'fanatics.' Users of the language C claim they 'will wipe it out.' Forth 'has no floating point math,' and while it may be 'useful for small programs, it is inadequate for large ones.' These criticisms are addressed below.

1) Write-only. Con: Anyone can write unreadable code in a dozen languages, including Forth.

Pro: It is easier to make sense of undocumented Forth than of undocumented BASIC. The usual complaint about Forth is its low redundancy, which is at the root of its differences from other languages. If 'nouns' are data, 'verbs' act upon data, and in FORTRAN, BASIC, and Pascal, the nouns are explicitly interspersed among the verbs. In Forth, the nouns tend to hide on the stack, making the program seem like a string of expletives.

Again, an ordinary language consists of repetitions of control structures, each enclosing a few lines of functional code. The programmer's eye converts the pattern of redundancy into the flow of control. The chief virtue of Pascal is its enforcement of this structure, making it easy for an instructor to detect a 'good' program by eye. In Forth, control structures are distributed one per word, and the unprepared simply cannot find a 'program' without reorganizing their thinking. But good Forth code reads much like (slightly stilted) English.

2) Language for geniuses. Con: Charles Moore may well be a genius, and the users of Forth are at the forefront of their fields (which include astronomy, textiles, primary education, image processing, expert-system creating, robotics, graphics, arcade games, 'Star Wars' special effects, hydrogen fusion, and the manufacture of navigation equipment, plywood, computer aided design tools, microdensitometers, computers, and Fourier-transform spectrometers).

Pro: An elementary class was split, with half learning BASIC first, the other half learning Forth first. Each half was then taught the other language. The responses were prompt and vocal: Halfway through the first session, the BASIC group was indignant that the others were so far ahead. The Forth group, when asked to learn BASIC, saw no point to it and did not.

3) Small user base. Con: Fifteen years ago there were only two Forth programmers.

Pro: Many of today's estimated 30,000 Forth users have come to it reluctantly and in desperation, because nothing else would

work for them.

4) C's industrial base. Con: With both Bell Labs and Digital Equipment Corporation behind it, C is assured of permanent support. Some C users are sure that this means the demise of all competitors.

Pro: C is not an interactive language, and so misses the principal advantage of microcomputers. Being optimized for DEC computers, C is as opaque as assembly language, and produces suboptimal code on other machines. Many professionals develop C and FORTRAN programs in Forth, then translate.

5) Integer arithmetic. Con: In Forth's original laboratory environment, all data were digital integers, and the language incorporates a number of features which optimize the speed and accuracy of integer arithmetic. Purists make use of integers an article of faith, 'in the spirit of Forth.'

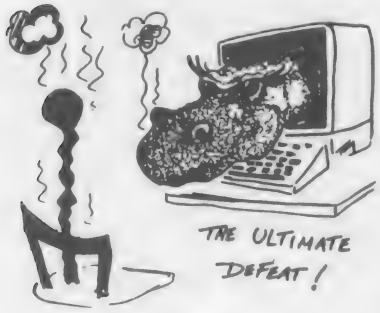
Pro: Moore advocated integers 'until there is hardware floating point' -- meaning the 8087 numerical coprocessor. MMS Forth is an adequate tool for computation-bound floating point number crunching, such as the Mie light scattering equations, which rational people do on supercomputers because they demand recursively calculated Bessel functions, Legendre polynomials, infinite series, complex arithmetic, 16-digit precision, and sophisticated graphics.

7) Useful only for small applications. Con: The forte of Forth is its ability to put a fast high-level language into the smallest possible space. In turn, it does not need many features customary in languages designed for megaprograms, and does not enforce typographic and stylistic rigidity a la Pascal.

Pro: For the applications envisioned in this article, size is not a stumbling block. However, General Electric's massive locomotive-repair 'expert system' was written in Forth for the usual reason: it was the best way to get the job done.

GIVE A BIT!!!

Contribute to the Newsletter this month.



(23)

Portland Atari Club

CHAIN LETTERS

by Kenneth J. Pietrucha - JACG

Every so often it happens. You open your mail and begin reading something which goes like this..."Add you name to the top of this list. Send $10.00 to the name on the bottom of the list and then remove this name. Make ten copies of this letter and sent them to ten friends. When your friends get this letter they will add their names to the top of the list and you will become the number two name. When your name reaches the bottome of the list, the people in position one will send you money ! Joe MpfAKI broke the chain and all his children were born bald."

Now, everyone looks for a quick way to get rich, but before you reach for your wallet, think about this for awhile. I have used an example where you only have to make ten copies, but I know that many chain letters ask you to make 25 copies. For the sake of this discussion, let's stick with my ten copy example.

You send the chain letter to ten people with your name in position one. If the ten people follow the same instructions as you did, then your name goes to position two and there are now one hundred chain letters in circulation with your name on them. When your name reaches position three, there will be 1000 letters. Don't forget, your objective is to reach the bottom of the list which is position ten. If you don't get to position ten, you don't get a dime and you can't quit your job.

Now comes the part I like. As I understand it, there are approximately 80 million households in the United states. If no one breaks the chain by the time your name reaches the eighth position, one hundred million copies of this letter should have been sent, which is 20 million more than there are households. Have you got the message yet ? There are not enough households on the face of the earth to allow you to get your name in position ten.

So, don't quit your job just yet. This is not the way to get rich...besides, it's illegal.

For fun, why not try to write a program to calculate the number of letters in a chain for a different number of copies.

(24)

Kushner Demos Forth

(Part One of Three)

by Donald Forbes - JAC0

No amateur is better equipped to present the Atari to other amateurs than Richard Kushner. Armed with a doctorate in chemical engineering, he does basic research on advanced semiconductor process control for Bell Telephone Labs in Murray Hill, NJ. In November 1981 he founded the Jersey Atari Computer Group and became editor of its monthly newsletter.

When he stepped down as president three years later the group had 550 members that filled the cavernous Bell Labs auditorium from month to month, the best monthly (28-page) newsletter in the country, a huge software library, an unmatched bulletin board, and $8,000 in the kitty.

When the Hayden Book Company needed a book on Atari BASIC, they put him in touch with James S. Coan who had previously written two books for them ('Basic BASIC' and 'Advanced BASIC') followed later by books on BASIC for the Apple and Commodore 64. Dick was determined to do the job right. This book would not be a pale copy of Basic for the Brand-X computer. The result was 'Basic Atari BASIC,' a complete guide to BASIC on the Atari.

Dick is not a FORTH programmer. We can only speculate on what kind of a demonstration he might stage for a FORTH audience. Here is one scenario, based on observations of his performances over a two-year period. His mastery of the Atari translated into FORTH will allow you to stage an imposing exhibit of the strong points of the Atari computer in data handling as well as its sound and graphics capabilities.

Data handling

We begin with the basics in an elementary introduction:

```
: FIRST-PROGRAM
  CR ." Here is an example "
  CR ." of a program "
  CR ." in Atari Forth." ;
```

The average of six numbers can be calculated with this program:

```
: AVERAGE
  36 45 65 89 91 56
  + + + + + 6 / .
  ." Average " ;
```

You can use a reverse slash to insert comments in your program with this code, which causes the compiler to skip the rest of the line:

```
: \ ( skip rest of line )
  IN @ 32 / 1+ 32 * IN ! ;
  IMMEDIATE
```

so that you can do this

```
: COMPUTERS \ Reverse slash comment
  \ This is all about computers
  ." This is all about computers" ;
```

Suppose we have a record of gasoline purchases for a brand-new car. This program will calculate the mileage for each tankful of gasoline.

```
0 VARIABLE MILEAGE1
0 VARIABLE MILEAGE2
0 VARIABLE GALLONS
: $IN CR ." ? " QUERY 1 WORD
  HERE NUMBER DROP ;
: GAS-MILEAGE ." First reading "
  $IN \ Input mileage1
  MILEAGE1 ! CR BEGIN CR
  ." Gals " $IN \ Input gallons
  GALLONS ! ." Mileage "
  IN@ \ Input mileage2
  MILEAGE2 ! MILEAGE2 @ MILEAGE1
  @ - GALLONS @ / . ." mpg "
  MILEAGE2 @ MILEAGE1 !
  0 UNTIL ;
```

Note that the word $IN works for signed integers like the INPUT statement in BASIC.

Suppose we wished to run the same program with stored data, rather than data entered from the keyboard. Then we can use this variation:

```
0 VARIABLE MILEAGE1
0 VARIABLE MILEAGE2
0 VARIABLE GALLONS
: $IN CR ." ? " QUERY 1 WORD
  HERE NUMBER DROP ;
DECIMAL 2303 VARIABLE DATA
127 , 4567 , 177 , 7094 ,
111 , 8955 , 138 , 11316 ,
: GET-GALLONS 4 * 2 +
  DATA + @ GALLONS ! ;
: GET-MILEAGE2 4 * 4 +
  DATA + @ MILEAGE2 ! ;
: MILES-PER-GALLON
  DATA @
  MILEAGE1 ! \ Input mileage1
  4 0 DO
  I GET-GALLONS \ Input gallons
  I GET-MILEAGE2 \ Mileage2
  MILEAGE2 @ MILEAGE1 @ -
  GALLONS @ / CR ." Mpg " .
  MILEAGE2 @ MILEAGE1 ! LOOP ;
```

To count numbers with a display, this program will work.

```
: COUNT-WITH-DISPLAY
  0 BEGIN 1 + DUP . AGAIN ;
```

To count to seven with a display we can use

```
: COUNT-TO-SEVEN
  0 BEGIN DUP 7 < IF 1 +
  DUP . THEN AGAIN ;
```

Suppose we have a relative who has promised to give us five times our age in dollars on each of our first twentyone birthdays. This problem can be solved with the logic of the counting program.

```
: BIRTHDAY-DOLLARS
  ." Total of $5 for each year "
  CR ." on each birthday "
  0 22 1 DO I 5 * + LOOP
  CR ." $" 4 .R
  ." after 21 years" ;
```

Here is another counting problem. You are the quality control inspector in a packaging plant and the average weight for five packages selected at random must be at least 180 grams. You want to write a program to ask the right questions and then accept the lot or reject it.

```
0 VARIABLE TOTAL
0 VARIABLE COUNTER
0 VARIABLE WEIGHT
: $IN CR ." ? " QUERY 1 WORD
  HERE NUMBER DROP ;
: PACKAGE-WEIGHT-MONITOR
  0 TOTAL ! 1 COUNTER !
  5 0 DO ." Weight " I 1 + .
  $IN \ Weight
  TOTAL +! 1 COUNTER +! LOOP
  TOTAL 5 / 180 < IF
  ." Reject this lot " ELSE
  ." Accept this lot " THEN ;
```

How do they get the computer to flip coins, deal cards or roll dice? All we need is the ability to generate numbers at random. Here is a way to generate ten random numbers. This program takes advantage of the fact that the Atari hardware generates a random number at byte 53770 in memory.

```
0 VARIABLE RND
53770 C@ RND !
: RANDOM RND @ 31421 * 6972 +
  DUP RND ! ;
: RND@ ( n1 - n2 )
  RANDOM U* SWAP DROP ;
: TEN-RANDOM-NUMBERS
  10 0 DO
  10000 RND@ CR 5 .R LOOP ;
```

We can now flip a coin 38 times which will just fill one line of the screen.

```
: RND@ ( n1 - n2 ) 53770 C@ SWAP
  /MOD DROP ;
: FLIP-COIN-38-TIMES
  38 0 DO
  100 RND@ 50 > IF
  ." T" ELSE
  ." H" THEN LOOP ;
```

To roll a die ten times we can generate a random number less than 60 and then divide it by ten to choose the face.

```
: ROLL-A-DIE ( ten times )
  10 0 DO
  60 RND@ CR DUP 5 .R
  5 SPACES 10 / 1 + . LOOP ;
```

Any program that requires input from the user is open to 'crashing' if the user inputs information that the program will not accept. Here is a way to protect the program against incorrect input.

```
: INPUT-PROTECTION BEGIN
  ." Pick a number from 1 to 20"
  #IN DUP DUP 1 < SWAP
  20 > OR CR IF
  ." Invalid input. Try again "
  ELSE ." Good guess! "
  ." That's my number too. "
  THEN 0 UNTIL ;
```

(Part two next month.)

ORNJUCE
Atari Computer Assoc. of Orange Co. (CA)

(25)

JACG Membership

The Jersey Atari Computer Group (JACG) invites you to become a member. Dues are $20.00 per year and entitle the member to: 1) Receive the monthly newsletter; 2) Purchase programs from the group's extensive tape and disk libraries at special rates; 3) Join special interest groups or form new ones; 4) Benefit from the expertise and experience of other Atari computer users. 5) Participate in group purchases of software at substantially reduced prices; 6) Receive a membership card that entitles the member to discounts at local computer stores; 7) Attend monthly meetings to learn about the latest hardware and software, rumors, and techniques for getting the most out of your Atari computer; 8) Submit articles and programs to the newsletter and give demos and presentations at the monthly meetings; 9) Participate in sale/swap activities with other members; 10) Access the JACG nationally famous Bulletin Board; and 11) Have a lot of fun.

If all of this sounds good to you send a check or money order, payable to JACG, to:

Ron Kordos
201 Lake Valley Road
Morristown, NJ 07960

Remember, receiving the JACG Newsletter is just one of the many benefits of being a member of JACG.

Lightning-Fast Forth

by Donald Forbes - JACG

Nov 85

How do you compare the speed of one micro with another?

The Sieve of Eratosthenes (find all the prime numbers between 2 and 8191 and repeat nine times) is a standard benchmark for micros.

The IBM PC/XT with an Intel 8088 chip running at 4.77 megahertz (cycles per second) can do it in 11.6 seconds. An IBM PC AT with an Intel 80286 chip at 4 MHz can do it in 3.71 seconds.

What runs it in only 0.339 seconds? You guessed it! Charles Moore's new chip with Forth embedded in the silicon.

For $3,885 you can buy from Novix Inc. of Cupertino CA their Beta Board incorporating their NC4000P 8 MHz chip, hook it to the serial port of an IBM PC or compatible, and execute Forth code at speeds up to 8 million instructions per second.

According to computer expert Earle Jennings, the 0.339 second timing is for real: "That figure is not a typo...One thing immediately apparent about this device was that it was faster than anything else I had ever encountered...The NC4000 is screamingly fast!..."

You can run up to eight concurrent tasks (a round robin multitasking feature) because the onboard hardware divides the stack memory region into eight segments. You can switch tasks in less than 5 microseconds.

The story goes back to October 1980 when John Peers, a robotics expert, was invited to join the board of Forth, Inc., which was founded a dozen years ago by Forth inventor Charles Moore. Don Colburn of Creative Solutions (they put Forth on the MAC) spent $1000 birthday present from his wife to investigate the merits of Forth on a chip and funded a one-day project organizing session with Charles Moore, Bill Ragsdale of the Forth Interest Group, and a chip design consultant. By March 1983 Chuck Moore demonstrated a color simulation of the processor.

By March of 1984 the Novix partnership commenced operation. It took four years and a million dollars to get to the detail design stage. It took seven months and seven hundred thousand dollars to put Moore's operational Forth processor on a chip.

This is not the first Forth chip, but it is certainly the fastest. Rockwell fit a good chunk of Forth into its R65F11 chip, along with the 6502 instruction set. The British Metaforth MF16LP single-board computer, which is implemented with custom bipolar circuits, uses Forth as its machine language. Bipolar technology is also incorporated in the H4TH/X Forth engines from Hartronix, which provide a real-time system with up to 4,000 primitives in firmware.

The press has begun to sit up and take notice. Electronic Design in their March 21, 1985 issue ran a story on 'Fast processor chip takes its instructions directly from Forth.' Novix reported that the "article put us on the map. Three hundred inquiries were developed. We found that Forth has friends throughout industry just waiting to show their management the opportunities." BYTE for October devoted a half page to the Novix Beta-Board.

(26)

Next came Computer Language with a ten-page article by Earle Jennings on the architecture and hardware aspects of the chip announcing that "Language-on-a-chip technology creates new programming frontier." Jennings shows in great detail how the Forth virtual machine was converted into hardware: he covers the registers, pins, op codes, bit fields, data and return stacks and all the rest.

Then Dr. Dobb's Journal it its yearly Forth issue carried nine pages on "A Threaded-Code Microprocessor Bursts Forth: Subroutines Without Performance Anxiety" by Leo Brodie of "Starting Forth" fame.

Brodie writes:" What does all of this mean to the programmer? It is easily demonstrated that the NC4000 runs faster than conventional micros. Because Forth instructions execute in a single clock cycle, the chip runs Forth code about 100 times faster than Forth running on a conventional processor. A benchmark using the Sieve of Eratosthenes reveals that the NC4000 runs Forth over 10 times faster than the 68000 runs its own machine code...Future revisions will increase the speed considerably...The real wonder of the Novix chip is that it allows execution of an elegant, high-level, modular language directly in the logic of the CPU."

He notes that Charles Moore, the creator of Forth and chief architect of the NC4000 processor, has claimed that the Forth chip represents "a landmark in the evolution of hardware and software."

John Golden, the guiding light of Novix, is anxious to tell their story. The document on the "Novix 4000P Forth System" by Greg Bailey has lots of important information. The only reference material available covering code generation (which is quite good) is the 49-page preliminary edition of the "Programmer's Introduction to the Novix NC4000P Microprocessor" by Leo Brodie. Novix (408/996-9363) is at 10590 N. Tantau Ave. in Cupertino CA 95014.

What next? Earle Jennings observes: "This the first time I have ever encountered a new machine architecture where there are already over 10,000 system programmers proficient in its assembly language and able to develop code on everything from a VAX computer to a Commodore 64."

A LOGO XMASTREE

SPACE PROJECT 12/85

by Ginny Smith

Merry Christmas from Ed and Ginny Smith to all SPACE Members

Ginny Smith, wrote a LOGO program that is in keeping with the season. She sent the program to the Newsletter editor to be printed and shared with everyone in the club.

If you have LOGO just type the program into the computer, save the program (SAVE "B:XMASTREE"), and run it (type XTREE(RET)).

For those who do not have LOGO, the picture will be displayed at the December meeting.

```
TO XTREE
BASE PU HOME FD 100
PD SETBG 112 SETPN 2
SETPC 2 90 LT 130
FD 25 L FD 15
R FD 35 L FD 15
R FD 45 L FD 20
R FD 55 L FD 25
R FD 62 L FD 190
L FD 62 R FD 25
L FD 55 R FD 25
L FD 45 R FD 20
L FD 35 R FD 15
L FD 25 PU HOME
FD 100 RT 90 FD 7
LT 90 STAR PU
FD 8 SETH 10 BK 15
TRIM FLICKER
END

TO BASE
CG PU BK 60 LT 90 PD SETPN 2
SETPC 2 90 FD 10 RT 75 FD 15
RT 105 FD 31 RT 105 FD 15
RT 75 FD 10
END

TO L
LT 140
END
```



```
TO R
RT 140
END

TO STAR
PU LT 90 FD 3 PD RT 100
SETPN 1 SETPC 1 127
SPI 5 144 1
END

TO TRIM
PU BK 30
RT 90 FD 10 C
RT 135 FD 30 C
LT 145 FD 50 C
LT 155 FD 50 C
RT 75 FD 50 C
BK 60 RT 90 C
LT 110 FD 80 C
RT 80 FD 50 C
RT 85 FD 50 C
BK 40 C HT
REPEAT 4 [PR "]
 PR [MERRY CHRISTMAS AND A HAPPY NEW
YEAR!]
END

              TO FLICKER
REPEAT 100 [SETPC 1 112 WAIT 20 SETPC
1 127 WAIT 20]
END

TO SPI :STP :ANG :INC
IF :STP > 27 [STOP]
FD :STP RT :ANG
SPI :STP + :INC :ANG :INC
END

TO C
SETH 0 PD SETPN 0 SETPC 0 7
FD 8 PU
FD 5 PD SETPN 1 SETPC 1 101
FD 0.5 PU
END
```


Learning Through LOGO

by Susan Wolff

Components of a Well-Balanced Computer Program

Although in the two previous issues I have been discussing LOGO and its applications in the classroom, there are many components of a complete computer program in the schools: a) history of computers, b) parts of a computer, c) how computers work, d) software, and e) a programming language.

History. You would be surprised, if you haven't been in a classroom lately, how many children when questioned about where computers came from, would respond, "Sears".

It isn't that children need to memorize dates or technical aspects of the computer revolution, but they should be given a sense of the developmental progression of automatic calculating devices over the centuries. They should get a feel for the idea that technology advanced as needs changed in the world and life got more complicated. Children should know that computers didn't just come out of a box, but rather are the result of other kinds of revolutions that occurred over the centuries. The development of the computer should be put into the perspective of other political, industrial, and social changes that went on in the world. And children need to use this historical perspective to think about the future and all the possible implications of this age of technology.

Parts. I believe that young people are capable of being comfortable with the vocabulary of today's technology. Not so that they can be tested on spelling or definitions, but so that they can carry on intelligent conversation about something very commonplace in today's world. I cringe when I hear a small child say, "Do I turn this thing on first?", or "Is this watchamacallit plugged in?". Using words like monitor, disk drive, or even central processing unit should be as natural to children as saying microwave or videotape. How are you going to be in control of a "whichery" if you don't even know its name? Children can learn what all the parts of a computer are and what goes on in each part. It's not suggesting that this learning include technology that is beyond the child's interest or ability to comprehend, but even very young children can at least use the correct vocabulary.

How They Work. Learning about how computers work is really tied to the parts of the computer. I will say again that teachers do not need to get too technical with the children, but certainly should encourage curiosity about what is going on inside the computer. A lesson on the binary number system can teach the children a lot about our own base ten system.

As a comparison, just as it is true that a child does not need to know what makes a ten-speed bicycle operate the way it does in order to utilize it, when that new bike arrives it is a wonderful opportunity for the whole family to find out about gear ratios.

Teachers do not need to get degrees in computer science to satisfy their class's curiosity about computers, but certainly the time when the class begins to use the computers again in the fall is a wonderful time for finding out what is going on inside the computer. There are many well done resources available for school libraries that would help even the youngest users get the general idea.

Software. Part of a well-rounded computer program would also have to include a variety of good educational software. I emphasize GOOD. I would rather see a school invest in a few pieces of quality software that are chosen with care, than go out and buy many mediocre programs. Today's software can provide children with a variety of problem-solving experience. I would also put, at the top of the list, an easy to use word-processing program.

Language. Last, but certainly not least, I would include a programming language, my first choice, of course, being LOGO. My reasons for choosing LOGO were enumerated in last month's article.

So ask your children what they have been doing with the computers in school. Volunteer an hour of your time and go into your child's classroom and share what you have been doing with computers. Make sure your children have been getting a well-rounded computer experience. If you don't have children you're welcome to come teach something to my classroom!

Although it is early in the year, I thought I would share with you the first program written by three of my fifth graders this week. They are just beginning to use LOGO, but this program will illustrate how they have started to use LOGO to integrate computers into our social studies program:

```
TO ENDING
TS CT
SETCURSOR [14 10]
PR [THE END]
END

TO PICI
HT
FD 40 LT 90 FD 30 RT 30 FD 50 RT 90 FD 10 RT 90
FD 40 LT 90 FD 10 LT 90 FD 60 RT 90 FD 15 RT 90
FD 60 LT 90 FD 10 LT 90 FD 20 RT 90 FD 10 RT 90
FD 30 RT 90 FD 10 LT 90 FD 40
PR [THIS IS A PICTURE OF A CACTUS.]
WAIT 200
END
```

[NOTE: in the listing below, boldface characters represent inverse video on the ATARI. Ed.]

(29)

```
TO PAGE3
TS CT
REPEAT 1 [PR[]]
PR [THERE ARE LOTS OF DIFFERENT PLANTS
IN THE DESERT, SUCH AS CACTUSES.
THERE ARE ALSO LOTS OF WEEDS.]
PR []
PR [PRESS RETURN TO CONTINUE.]
PR RL
END

TO CREDIT
REPEAT 3 [PR[]]
PRINT [JASON, JENNIFER, MELISSA]
PR []
PR [PRESS RETURN TO CONTINUE]
PR RL
END

TO TITLE
TS CT
REPEAT 9 [PRINT []]
PR [NORTH AMERICAN DESERTS]
END

TO DESERT
TITLE
CREDIT
PAGE1
PAGE2
PAGE3
PIC1
ENDING
END

TO PAGE1
TS CT
REPEAT 4 [PR []]
PR [NORTH AMERICAN DESERT]
PR []
PR [NORTH AMERICAN DESERTS ARE VERY
DRY.  WHEN YOU THINK OF ESERTS YOU
THINK OF CACTUSES.  DESERTS ARE VERY
NOT MOST OF THE TIME.]
PR []
PR [PRESS RETURN TO CONTINUE.]
PR RL
END

TO PAGE2
TS CT
REPEAT 4 [PR []]
PR [WHERE DESERTS ARE FOUND]
PR []
PR [DESERTS CAN BE FOUND IN MANY
STATES.  SUCH AS: NEW MEXICO, ARIZONA,
UTAH, NEVADA, CALIFORNIA, AND PARTS OF
OREGON.]
PR []
PR [PRESS RETURN TO CONTINUE.]
PR RO
END
```

pour. I can run an IBM PC at work after hours, using public domain Forth. I can run Wang BOBO Forth with floating point at work after hours. The machine is designed for business applications instead of games and graphics. Is there any user group support?

CONCLUSION

Would it not be wonderful to enjoy the graphics of the Amiga at the price of the 520ST supported by a public domain Forth (like the F83 for the IBM PC) that would be welcomed by the JACG membership, and to have Analog Computing and Antic magazines both clamoring for coverage? Perhaps next year? Or the next?? Or the next???

What Good Is MIDI?

by James Miller - JACG

NOV 85

The JACG newsletter was first to mention MIDI to many of us users and also gave us info on the Hybrid Arts interface for the 800 before the ANTIC article was published. Since my last article we're starting to hear more about MIDI. Unfortunately, it's the new design computers like Amiga and the ST that are addressing this development, leaving us 800 owners thinking about buying a new system. Regardless of the computer, it's time to think more about applications.

Some questions I've come across deal with controlling drum units from your synthesizer keyboard. Yes, individual drum sounds like snare, kick, toms, etc. only require one channel. Where the drums are layed out C-D-E-F-G-A-B on the keybord. MIDI will address up to 16 channels simultaneously. But you are losing computer memory to drum data where it could have been used on synthesizer over-dubs. Many Drum units have their own memory for patterns and even entire songs. All you need from MIDI is a TRIGGER or timing pulse to keep all devices running at the same tempo. Synthesizers typically have small sequencing memory and are not designed to play songs from start to finish. So this is where you should put the computer to work for you.

It seems reasonable to believe that future systems will incorporate MIDI buffers much like printer buffers that will allow the software the freedom to be more powerfull in editing during play. This allows for disk access time to replenish the buffer and room to incorporate windows. One window may monitor MIDI events and another window would be a word processer for writing lyrics concurrently. Then after a productive session the finished score complete with words can be printed out in sheet music form.

Another use of a MIDI buffer could be a high quality jam session over the modem with other musicians. Text would be mixed with MIDI data on your terminal. No more loading up the van with amps and going to rehearsal. Best of all, to "tear down" you need just to drop carrier and play back or maybe edit the final mix of the entire session that your software was downloading to disk as your "Band" rehearsed. Then you print the sheet music.

(30)

Among some new products that are now available in music stores, is a MIDI device using a microphone or guitar as input. It analyzes and translates this analog input into digital MIDI event information output. You'll still need a synthesizer to provide the sound, but you don't have to learn Keyboard to make it all happen. So to answer my first question 'what good is MIDI ?', A: It could be real good, just hang in there.

Mathematics 01
Mathematics (1)

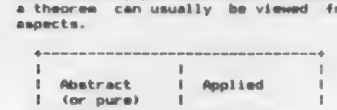
11-85

If you would master mathematics, the language of science and the apotheosis of our culture, you must strive to be a mathematician's mathematician.

You will need to master the conceptual structure of the mathematical sciences in general, and of abstract mathematics in particular.

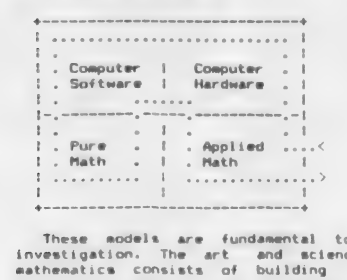
Until 1950 the conceptual structure of the mathematical sciences could be represented simply as a rectangular black box, with the left half representing abstract (or pure) mathematics, and the right half applied mathematics. The user with a problem invoked the applied mathematician who in turn invoked an algorithm provided by the abstract mathematician which returned the answer to the applied mathematician. He, in turn, supplied the solution to the user.

Note that in this model the left and right halves served as mirror images of one another, like the obverse and reverse of a coin. Duality pervades most of mathematics: a theorem can usually be viewed from two aspects.



Installation in 1950 of the first commercial computer made this model obsolete. The two part rectangle became a rectangle with four quadrants, with computer hardware in the first quadrant, computer software in the second quadrant, abstract mathematics in the third and applied mathematics in the fourth.

The path of problem solution now moved from the applied mathematician to the computer hardware to the computer software to the abstract mathematical algorithm and back to the problem proposer. In this extended model, the vertical axis became the axis of applications about which the two halves were mirrored, and the horizontal axis the axis of computers, about which computer software and hardware mirrored the mathematical structures.



These models are fundamental to this investigation. The art and science of mathematics consists of building models which can be applied in other contexts. This inquiry uses mathematical models to build an overall model of the mathematical sciences as one integrated unit, and then uses analysis and synthesis to break down the components and reassemble them into a coherent and comprehensible structure.

(31)

The essential unity of mathematics has been stressed by many investigators. In the words of Professor Lothar Kollatz of the University of Hamburg: "The author would be delighted to find that this book contributes to showing how absurd the distinction between 'pure' and 'applied' mathematics actually is; there is really no boundary that separates the two. There is only one mathematics, of which analysis, topology, algebra, numerical analysis, probability theory, etc., are merely some overlapping areas."

The next step, and the crucial one in this inquiry, is to determine the conceptual structure of abstract mathematics. Here lies the heart of the whole inquiry. The structure that will be established for this quadrant will then serve as a model for the structures in the other three quadrants, and thus preserve the integrity of the whole.

Structure of abstract mathematics

A preliminary structure for abstract mathematics can easily be constructed by drawing a Venn or Euler diagram with three overlapping circles for the three principal domains or dimensions of mathematics, labelled in turn 'geometry,' 'algebra,' and 'analysis,' where the term 'analysis' is used in the limited sense of the differential and integral calculus.

The diagram provides a set of pigeonholes for the intersections: geometry and algebra, geometry and analysis, and geometry and algebra and analysis.

One critical dimension in this analysis is the time dimension, often neglected in mathematics, to which we will return in greater detail. The time dimension is important only in the sense that it illuminates the present: How did we get to here from there?

We can now order the components of the Venn or Euler diagram in chronological sequence: 1. geometry; 2. algebra; 3. geometry algebra; 4. analysis; 5. geometry analysis; 6. algebra analysis; 7. geometry algebra analysis.

Although this simple model shows internal consistency, a moment's reflection shows

(32)

that it is incomplete because it fails to provide space for the topology of point sets and other consequences of the investigations of Georg Cantor.

Expansion of our original Venn or Euler diagram to four circles produces an unsatisfactory result, adding complexity where simplicity is needed and desired.

The solution is to discard the circular diagrams and replace them with a Veitch diagram or Karnaugh map of four dimensions: geometry, algebra, analysis, and topology of point sets. The eight cells of geometry cluster around the vertical axis, of algebra around the horizontal axis, of analysis in the top half and those of topology in the right half.

We can now order the intersections in chronological sequence as follows: 0. null set; 1. geometry; 2. algebra; 3. algebraic geometry; 4. analysis; 5. geometric analysis; 6. algebra analysis; 7. geometry algebra analysis; 8. topology; 9. geometric topology; 10. algebraic topology; 11. geometry algebra topology; 12. analysis topology; 13. geometry analysis topology; 14. algebra analysis topology; 15. geometry algebra analysis topology.

We can use a binary (or hexadecimal) numbering scheme for these intersections using four bits: 0001 for geometry, 0010 for algebra; 0100 for analysis; and 1000 for topology. Thus the intersection of all four would be labelled as 1111. The numbering scheme also preserves the historical sequence: 0000 for the null set, and 1111 for the intersection of the four dimensions.

		topol	topol
analy	analy	analy	analy
4	geom 5	geom D	C
		topol	topol
analy	analy	analy	analy
alg	alg	alg	alg
6	geom 7	geom F	E
		topol	topol
alg	alg	alg	alg
2	geom 3	geom B	A
null		topol	topol
0	geom 1	geom 9	8

To understand mathematics today one must understand its origins. The Karnaugh map or Veitch diagram provides a set of pigeonholes where the advances can be classified in chronological sequence. More importantly, the diagram provides a means of identifying and cataloging discoveries or inventions that are replications of earlier discoveries in a different guise. As the quadrants of our earlier rectangle mirror one another, so do the cells of the Veitch diagram mirror one another.

History of mathematics

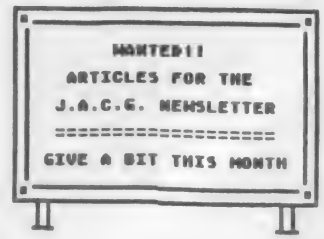
The history of mathematics must be viewed in terms of paradigm shifts: new discoveries that caused a 'revolution' in the way that past mathematics was observed. The dimensions of our Veitch diagram or Karnaugh map are designed to conform to these paradigm shifts, and the boxes are designed to accommodate them.

In the broadest possible perspective, there were four events that forever changed the course of mathematics. First was the publication of Euclid's Geometry around 300 B.C. Second was the discovery of algebra by al Kwaharizmi and the publication in 1545 of the "Ars Magna" by Geronimo Cardano. Third was the publication of Newton's Principia in 1687. And fourthly, the publication of Cantor's book on Transfinite numbers.

Within these developments were others that fill in the intersections of the diagram. The box for the null set serves as a repository for the investigations into logic started by Zeno of Elea in 600 B.C. The intersection of geometry and algebra is properly marked by the publication in 1637 of the appendix on geometry in the Discourse of Method of Rene Descartes. The intersection of geometry and analysis is marked by the publication by Karl Frederic Gauss of his volume on differential geometry entitled "General Investigations of Curved Surfaces." None of the later intersections, however, are as clearly marked.

We could attempt to fill in the boxes with the names of individuals who did outstanding work in each different area. This exercise would have some pedagogical interest, but lacks the precision needed for the rest of this inquiry.

A brief outline of the essentials of the history of mathematics (needed to fill in outlines of the model above) will be presented in the next chapter of this inquiry into the internal structure of the mathematical sciences.



ODDS AND ENDS

by Sharon Brown, ACAOC OrnJuce

As promised last time, this month we'll continue looking at structured programming and WHILE-loops. However, I'd first like to take a slight detour. In a course I'm taking at Rancho Santiago College, I'm using a series of programs written to supplement a calculus course. While generally quite helpful in terms of learning calculus, these tutorials leave a lot to be desired in terms of user-friendliness.

Aside from not giving the user a chance to correct erroneous input (a problem of all too many programs, unfortunately), two types of user-unfriendliness stand out in the tutorials. One type results from an attempt to be "clever." The main disk menu is displayed with a cursor moving in a continuous loop past the names of the programs on the disk, pausing briefly beside each one. This looks very slick and classy, but it is ridiculous to use. To select a program, the user must wait until the cursor is beside the desired program, and then hit [RETURN] before it gets away and moves to the next title. Woe to the user who has slow reflexes!

The second unfriendly feature is in a program intended to graph any equation. It results from the assumption that the user is also a programmer (which isn't true for many students in the class.) The program halts execution after asking the user to add a program statement containing the equation to be graphed. The necessary format of the statement is described rather sketchily on the screen. However, the instructions for the statement neglect to mention that the user must type RUN to continue execution after the equation is entered. A programmer would probably figure this out, but a non-programmer would have a devil of a time with it. Hyper-cleverness and lack of consideration for less experienced users are both programming faults that can be easily avoided.

Back to the subject of WHILE-loops. I must confess that I have yet to work out a sample program to illustrate the concept. Therefore, I'd like to spend the rest of this column discussing classes of exit conditions for such loops. Naturally, any expression, no matter how arcane or complex, that can be evaluated as true or false can be used to control the exit from a loop. However, there are certain classes of conditions that are particularly useful.

In almost any kind of data-processing computer application, programs will contain loops which read or input data records one at a time, processing each in turn. Typically, such a loop will terminate when there is no more data. Since Atari BASIC does not provide an explicit test for the end of a file, a sentinel or trailer value or record is generally used to indicate the end of the data. The program will then exit the loop upon encountering the sentinel value. To accomplish this, a value/record must be read just before each execution of the loop exit test. In BASIC, as opposed to more structured languages, there are two possible ways to structure such a loop, as shown in the diagrams below.

The first method is the "textbook" method, which uses a "priming read" to give the exit test something to test before entering the loop proper. It is a carry-over from languages like Pascal in which the loop has to start with the exit test. It has the advantage of emphasizing the loop structure and conditions. In BASIC, however, the shorter second structure will also work.

Now, suppose you wish to process a set of records to the end UNLESS some particular condition is encountered in processing the data. The simplest way to code the test for this condition is to simply branch out of the loop as soon as it is encountered. Unfortunately, such an approach creates code that is hard to read and update, as this second condition is not readily apparent. A more readable approach is to use a "flag" variable which is set to a certain value if and when the second condition is encountered. The flag then functions as a logical variable that can take on one of two values, one representing "true" and the other "false". The flag is initialized to the "false" value before the loop and is set to true only if the second condition is encountered. The exit test then checks for either of two conditions, exiting the loop either when it runs out of data or when the flagged condition has become true. Several flags can be used to control the same loop. If they are given meaningful variable names, they can make the exit conditions for the loop quite clear.

A third common type of exit condition is the counter, a variable that is incremented each time through the loop until it reaches a pre-determined value. Counter-controlled loops are not commonly used in BASIC because they serve the same basic purpose as FOR-NEXT loops. Since the BASIC FOR-NEXT loop is quite flexible, it is usually all that is needed. However, under some circumstances counter-controlled WHILE-loops may be superior in clarity to FOR-NEXT loops. Once again, the problem centers around having a second condition that may cause the loop to be exited prematurely. Although it is legal to branch out of the middle of a FOR-NEXT loop, it is rather sloppy. The secondary exit condition is again buried in the middle of the loop and is not apparent to the person reading the code. In such a case, a counter-controlled loop in which the value of the counter can be tested in the same statement with other conditions for exiting the loop provides superior readability. OJ

Looking Over Some Shoulders

by Kevin McGonagle, ACAOC OrnJuce

Salt shaker time again...

IT'S GETTING SO 64K JUST ISN'T ENOUGH... The September issue of Byte magazine has an article detailing how to upgrade an Atari 800XL to 256k, and how to get software to use the memory enhancement. Pretty good for a magazine that usually designates the Atari as a "game machine".

WHILE ON THE SUBJECT OF MAGAZINES... Infoworld's review of the 520ST is chock full of contradictions. It was called a review of the machine, but most of the article was one of bemoaning the lack of software. The article praised item after item about the machine, but then gave it an overall rating of poor. Hardly the computer's fault about lack of software.

WHILE ON THE SUBJECT OF ST SOFTWARE... Finally the boxes are filling up. Word processors, compilers, financial programs, just about anything you would want is here now! Even games for those game playing junkies. I'll put in a good word for the Floppy Disk in Downey and Authorized Computer Service in Foothill for the Atari ST software support. [Also The Sound Room in Anaheim and Learning Tree Computers in Tustin. ED.]

INFOCOMANIAC... They finally did it to some home computer owners. Infocom has released a game that won't run on the Atari/Commodore 8-bit machines. This one is written by Steve Meretzky, author of Planetfall and Sorcerer, and co-author of The Hitchhiker's Guide to the Galaxy, with Douglas Adams. It's called, A Mind Forever Voyaging, and will only run on IBM, Apple, and Atari ST machines. Infocom's dilemma always has been the 90K Atari disk drive and now they've done something about it.

AMIGA?... I expect to see a real showdown between the Amiga and the Atari ST, but the Amiga will have to show up first. It's getting pretty late in the year and if Commodore expects to sell enough units to pay for that fancy (and funny) advertising, those machines better hit dealer's shelves in big numbers soon. By the time the Amiga gets into circulation, the ST could have as much software as the Mac.

ENGLISH SOFTWARE IS GETTING THERE... That British software manufacturer is best known for games that are so hard as to be unplayable. Their latest game, Chop Suey, breaks that mold and is a winner. The karate simulation can be one player or two, and if you are looking for help with strategy, you can watch the program play itself. Look for this one, you won't be disappointed. And yes, it's for the 8-bit line.

BUSINESS, BUSINESS, BUSINESS... For those people that really know legitimate business oriented software, the alternate operating system for the ST was an exciting announcement. BOS (British Operating System) will give the new Atari ST many things, including some IBM compatibility and a higher level of respectability. Now the RAM based OS looks like a good move.

A FORTNIGHT'S WORTH OF WORK... Because GEM Write is taking too long to reach the market, Atari's John Feagans supposedly wrote ST Writer for the ST in TWO weeks. It should be out in October. [It's out now. ED.] Included is the ability to use files originally written in AtariWriter on the 8-bit line. If this product gets here before AtariWriter+ for the 130XE then we all have license to complain.

IBM'S FIRST MISTAKE... IBM hoped that executives, after sitting behind their big desks at work flailing away at their PC's all day, would want to go home and sit behind a PC Jr. Needless to say it didn't work. Maybe the opposite will though, after seeing how much the kids can do on the family's Atari, the power user might just buy an ST or two for the office. Don't laugh, Apple is using this strategy for the Macintosh, aren't they?

SOFTWARE'S ON THE IMPROVE... It used to be that we Atarians had tons of software titles to choose from, but not many were worth consideration. Nowadays the numbers are down but the quality is way up. With Kennedy Approach, Chop Suey, Hacker and Spellbreaker among the new programs to choose from, the choices are all good.

USE TV
TWO COMPUTERS

by BOBBY HERMAN

WAND NOV85

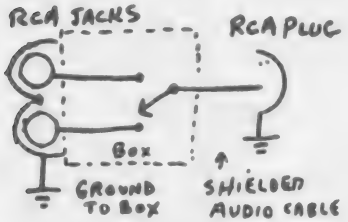
Many of our users have more than one computer. Some have an 800 and an 800XL or perhaps an 800 and a 130XE. Some of the older software will not run on the newer models without the use of translator disks, and some of the newer software being written for the 130XE does not run on the 800. Sometimes children use one model and the adults use the other. So many of the users have both computers on the same desk or table, and use one TV screen by plugging and unplugging the RF cable that goes into the TV/Computer switchbox. This becomes a nuisance.

I have devised a little inexpensive piece of hardware that will eliminate the substitution of the cables. It is a simple little switch box. The parts needed are available from Radio Shack as follows:

1. small metal box 0270-235 $1.69
2. SPDT switch part 0275-613 $1.69
3. 2 RCA phono jacks 0274-346 $1.79
4. 1 RCA phono plug 0274-339 $1.29
5. about 6 inches of shielded audio cable, soldering iron, solder, etc.

Wire up the circuit as per the diagram below. It is important to ground the outside terminals of the RCA jacks to the box and to ground theshielding of the audio cable to the box and to the RCA plug.

Plug the RF cables from the two computers into the RCA jacks and plug the RCA plug at the end of the audio cable into the TV/Computer switch box. When the SPDT switch is in one position the TV will read from computer number one, and in the other position from computer two without switching cables. I used a Dymo label maker to indentify the positions of the switch. It works very well and saves wear and tear on the cables and on one's nervous system.()



PEEKS AND POKES

by Kenneth J. Pietrucha - JACG

OCT 85

(35)

I am always on the look out for new PEEK and POKE locations: The longer you own your ATARI, the more difficult it becomes to find a truly new location.

At the last meeting Frank Pazel passed along a newsleter with some PEEKS and POKES from the Lawrence Atari Computer Club. One of the more interesting locations allows you to modify DOS to let you use lower case letters and punctuation marks in filenames.

To modify DOS, do the following, 1) have a new formatted disk ready, 2)from basic, POKE 3018,33:POKE 3022,129 and 3) go to DOS and use option "H" to write DOS to your new disk.

I tried this and it worked fine. I set up a 2 line dummy file and used lower case letters in the file name. The only problem I had was when I tried to rename the lower case filename with a filename using upper case, I got an error, twice.

In your travels, if you do find a new or odd PEEK and POKE location do pass it along to me.

Until next month...

PAPER CLIP DUMP

by BOBBY HERMAN

(36)

WAND NOV85

Paper clip is the 'hot' new word processor that many of us are now using. There have been lots of reviews of many of the various features of the program. One feature that is often skipped over lightly is a real sleeper. It is the ability to insert graphics into a text document.

Herman Silver and I worked on this one afternoon a few weeks ago. What we found was quite interesting and very useful.

First we discovered that there is a file on the PaperClip disk, called HIRESDMP.BAS. This is a program written in BASIC which can be used all by itself as an independent screen dump without PaperClip. It has drivers for Centronics, Epson MX and FX, Nec, Okidata 92, Seikosha AT100, and Anadex DP9500 printers. It also contains an option to create your own drivers for other printers. It will load and screen dump graphics from Koala and Atari touch tablets, Atari Light Pen, Syntrend, B/Graph, Fun With Art, and Atari Paint. It allows changing the shade of gray (for a black and white printer) of each of the four color registers from all white to all black and six in-between grays. The picture is dumped vertically and is of one size only, 5.25 inches wide, and 2.625 inches high. The picture does not come out centered on the page from left to right, but extends from column 15 to column 69 on my Epson MX80. The size and centering is slightly different on the Epson FX.

If you should decide to place this file on a separate disk be sure that the printer driver goes with it. The printer driver has to be in Drive 1 when the program is used.

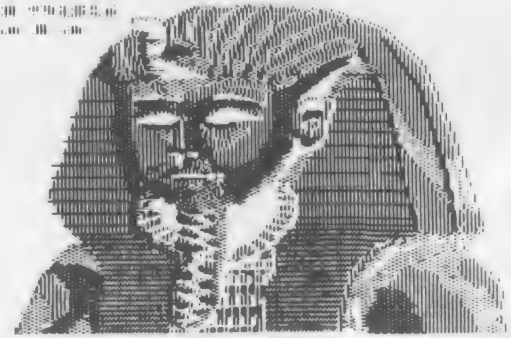
The use with PaperClip works very well. The graphic is loaded in, the shades of gray modified if desired and then an output data file is created from the graphic. When writing the PaperClip text, the following code is written at the place that the graphic is wanted and the whole thing prints out very nicely: CTRL ? VD2:FILENAME.EXT. One line is skipped on top and one on bottom of the picture. Also, there are two sample picture files on the disk. One is a Koala compressed file picture of King Tut, and the other is a 66 sector B/Graph.

This program, whether used with or without PaperClip, is a universal graphic loader and screen dump and is a useful addition to our graphic utility software collection.

Below is a sample of how a letter might look with a graphic in the text. The picture is the KING TUT that is on the PaperClip disk. This was not done by cutting and pasting, but by using the technic as outlined above.()

Dear Sir:

This is a test to see if this confounded computer and program will do what it is supposed to do.



This is the bottom without skipping a line.

(37)

Prometheus ProModem 1200
by Jim Rothrock, ACAOC OrnJuce

I was recently given the following assignment at work: find a good 300/1200 baud modem that would connect one of our terminals to a remote minicomputer. Thus began the Modem Hunt. After reading many advertisements and modem manuals and making quite a few phone calls, I finally found what I was looking for. In the $200 to $300 price range, the Prometheus ProModem 1200 is the best 300/1200 baud modem around.

The ProModem is a direct-connect modem which interfaces with your computer through a standard RS-232C port. It communicates at baud rates between zero and 300 baud, as well as 1200 baud. Other features include easy to read status indicator lights, auto dialing, auto redial when a busy signal is encountered, auto answer on any ring, and support of the entire "AT" command set, making the modem Hayes compatible. The ProModem even has extra commands which do things like printing help menus on the screen (listing all of the various commands) and printing out the current date and time from the modem's built-in clock/calendar. DIP switches on the bottom of the modem allow you to set defaults, such as whether auto answer is on or off, whether error codes should be displayed as numbers or in English, etc. The instruction manual which comes with the modem tells you how to set the DIP switches so that terminal programs will think that the ProModem is a Hayes Smartmodem.

In addition to offering a superset of the Hayes command set, the ProModem offers greater hardware capabilities than Hayes. Extra memory may be added, up to 64K, which allows you to store telephone numbers within the modem itself. Text files may be downloaded into or uploaded from this memory without any intervention from your computer. Also, an alphanumeric display may be added to the ProModem, which can tell you things like whether someone is currently on-line and when they connected even if your computer is off. These advanced features are an obvious advantage to users who wish to run a BBS on the ProModem.

The ProModem has other nice features. On some modems you must press a button to put it in 300 or 1200 baud mode. The ProModem simply sets itself to the baud rate coming out of your computer's RS-232C port (on the Atari, this would be the RS-232C port on your 850 interface or R-Verter). You can also use the "buffered mode," in which the modem communicates over the phone lines at 300 baud while communicating with your computer at 1200 baud. This may not seem very useful, but where I work it is difficult to change the terminal baud rate to 300. Buffered mode allows me to leave the terminal at 1200, but go over the phone lines at 300. Sometimes this is necessary due to line noise. Also, when using the auto dial command you may place the letter "W" within the digits of the phone number. When the digits to the left of the "W" have been dialed, the modem will then wait for a dial tone before dialing the digits to the right of the "W". This is a great help in using long distance services such as MCI and Sprint.

Also worthy of mention is the fact that the ProModem has a built-in speaker. This is so you can hear what is actually happening (number being dialed, ringing, busy signal, etc.). It has an adjustable volume and can be disabled completely with a command. The speaker isn't really necessary, since messages such as "RINGING" are displayed on the screen, but it is still helpful in situations such as a person answering the phone rather than a modem.

I have found that the ProModem is reliable, easy to use, and has advanced features not offered by similarly priced modems. If you are thinking about getting a 1200 baud modem, I believe that the Prometheus ProModem 1200 is the one to choose.

OJ

(38)

RAMBO/TERMINATOR-XL ARTWORK

By Mike Redmond

We have included some circuit board artwork in this ish for building either the RAMBO XL or TERMINATOR XL. The board is dual purpose. In fact, if you look closely, it also includes some unlabelled connections so that you hardware hackers out there can build what we call the DEEP THOUGHT XL of up to 2 MEGA BYTES....(and we wonder why some people call us crazy).

The circuit board artwork is NOT in an adequate final form (the copper paths are very thin) and it is NOT scaled for simple reproduction. I recommend that local clubs attempt to find someone that can convert this stuff into a final, dimensionally accurate transparency for circuit board production. Within a month, folks in MAAUG will have accurate transparancies that they may be willing to sell to some interested users groups. Indicate your interest by sending a letter to MAAUG. I doubt that MAAUG will get into distributing the board on any nationwide scale.

Please note that things differ a bit from info in Paul's construction article. Multicharacter on-board labels have been converted to single character labels with an external legend. The chips are not marked on the board in this artwork. The board also reflects a change in the design for the TERMINATOR XL circuit. See the TERMINATOR construction article for details.

ENJOY!...MR

MY ATARI 800-XL HAS HAS BEEN RAM-BOED!

By Paul Schnettler

I was thinking of buying another disk drive for my 2 year old 800XL. We all know how handy an additional storage device can be. Well, the money just wasn't there so I had to come up with another way...

After reading an article which appeared in BYTE magazine I became interested in the idea of upgrading the RAM which is resident inside the 600XL. The article described a way in which any brave 600XL owner could make a few modifications to their machine to get a quarter of a mega-byte of RAM. Well, that got me going and I'll tell you how I turned an ordinary 800XL into a super "RAM-BOed" 800XL. You can do it too...

First, credit where credit is due, Kurt Grittner and Mike Redmond got this whole project off the ground, their help was invaluable. Dave Mullenix fabricated the PC board, and Al Divine did the artwork for the prototype PC board. Thanks again guys!

I eagerly awaited the arrival of all the necessary parts and chips that I needed for the upgrade. The RAM chips and a few low power TTL chips, a DIP header socket, a few chip sockets, and a couple of caps and some wire. The only tools required were a low power soldering iron, a pair of small pliers, a small phillips screwdriver and a pair of steady hands. Oh yeah, don't forget a peice of tin foil for covering the table top that you work on, we don't want to ruin the new RAM chips by static discharges.

After I got all my stuff together it was time to open up the computer. I turned it over and removed the 6 phillips screws and carefully flipped the machine back over (keyboard side up), being carefull not to let the two halves come apart just yet. The 800XL keyboard is connected to the main board by a rather cumberscome flexible membrane cable/connecter arrangement which must be carefully removed from the connector. Don't try it with anything but your fingers.. A small solderless connector has to be unplugged from the cartridge slot area also.

The next thing I did was to remove some more screws which held the board to the bottom of the plastic shell, there were three of these screws along the back of the main board very close to the metal shielding. There was one more screw to replace located between the joystick ports. After removing these screws I was ready to remove the circuit board and get down to work. I carefully coaxed the snug-fitting board from the shell and proceeded to straighten the tabs which held the shielding to the circuit board. After I had straightened all the tabs I took a small screwdriver and removed the shielding. There it was, the guts of my 800XL. It didn't seem like there was enough room to fit anything more in there, but where there is a will there is a way. I was willing to go on.

I wanted to check the ANTIC chip to make sure I could modify my machine, so I checked chip U7, the video controller. This chip had to have the number C021497 to work correctly. I was in luck -- there it was.

NOW came the TRICKY part! Changing the RAM chips. I made sure the foil was under everything that I was going to touch with my hands during the rest of the procedure, including the board and the memory chips. I carefully pried the eight RAM chips from the sockets located along the left side of the board. I also removed the chip from the socket labeled U27 just to the right of all the RAM chips. Just behind U27 is a small 3 inch square area with relatively few parts. This is an area where the PC board would fit nicely.

I replaced the existing 64k RAM chips with the new 256x RAM chips, again being extreemly carefull not to ruin the chips with static discharge. I took the old chips and repacked them in the tube in which the new RAM chips had arrived. There was no use in trashing them since they might be usefull later. Next it was on to the fabrication phase.

I had to assemble the PC board, which was a bit tricky in it's own right, since the board is so small. The first thing to do was to cut the DIP header into two halves and trim away almost all the plastic from the header part. Once this was done I could solder the header pins to the pads on the board marked 'socket'. There are eight pins on each side of a chip and since there are two sides on each one it was a bit tedious (I've got to cut down on coffee!). Once that part was done, all the rest of the soldering was easy. I trimmed the header pins from the top side of the board (the non-copper side), and then soldered the IC sockets to the board. These sockets are optional but I wanted to be able to remove the chips easily in case of failure. This might be a problem in later models of the 800XL with different types of shielded cases which don't have the required height. The next thing I needed to attach to the board were a six wires and a jumper between two points on the board. Next I plugged the chips into the sockets assigned to each of them (they are clearly marked on the board) and soldered the wire coming from the pad marked RAS to the right side hole of resistor marked R32. This hole should run to pin 1 of each of the RAM chips. You will have to desolder the right side lead from R32 (even though it works without it), to be totally legal. (You should solder a 33 OHM resistor into this hole and then solder the RAS wire to the dangling end of this resistor...M.U.A.). Wrap the R32 resistor with some tape or remove it completely. The other five wires must run to the B port of the PIA chip which the 600XL doesn't use (the 2nd two joystick ports which the old 800's have built in). I had to rig up a connector to plug onto these pins made from a dip socket cut in half and trimmed to 5 holes for the five wires. The PIA chip is located at U23. It must be carefully pried from it's socket. Once the chip (labeled 6520) was removed, pins 12,13,14,15, and 16 had to be bent up so they stuck straight out and away from the chip. Finally the chip was carefully pushed back into it's socket.

The last part of the modification was connecting the wires to the IC socket correctly and plugging it onto the pins coming from U23. The correct pin connections are:

from: PB2	to: pin 12
PB3	pin 13
PB4	pin 14
PB5	pin 15
PB6	pin 16

Finally I was done soldering and the only thing I had left to do was plug my new PC board into the empty socket at U27 and connect the modified socket to the PIA chip at U23. I was now ready to see if I would blow anything up!

The quickest test would be to see if the machine at least came up in BASIC since it has its' own internal test routine. I plugged the power in and turned on the monitor, and then I switched the board on and patiently waited for those wonderful words "READY"... And there it was. I had done it! Now all I had left to do was to carefully reassemble the RF shielding over the board sides while making sure the new PC board was not going to short circuit on the metal covered shielding. Everything fit snuggly back into the computer's shell and I only had to carefully re-attach the keyboard cables before I was ready to finally close up the casing of what was and would remain a normally operating 600XL (until I got Kurt Grittner's new RAMBO-XL software).

Actually, the computer would act like a 130XE when the RAMDISK.COM file was loaded from DOS 2.5.. This is exactly what the extra board is supposed to do-- access the extra memory and use it for a RAM disk. But isn't something fishy here? The 130XE only has 128k of RAM and our upgrade has twice that. The 256k RAM I installed couldn't be used fully with an ordinary version of DOS 2.5. The new, special patched version was being developed by K.G. and so I had to wait.

Well the wait was not long and it was worth it. Now all I do is load up DOS 2.5 and I get drive 1 with 707 free sectors. In addition I have a drive 7 with 707 free sectors and also a drive 8 with 628 free sectors (it needs a few sector for some DOS files). Let's see now... thats 2034 free sectors to play with. I don't need another disk drive and I've got RAM disk space thats 50 times faster than a normal disk drive when it comes to copying and transferring files. The only disadvantage I might have is in making sure my files are saved to a disk or a cassette before I power down my computer (turn it off).

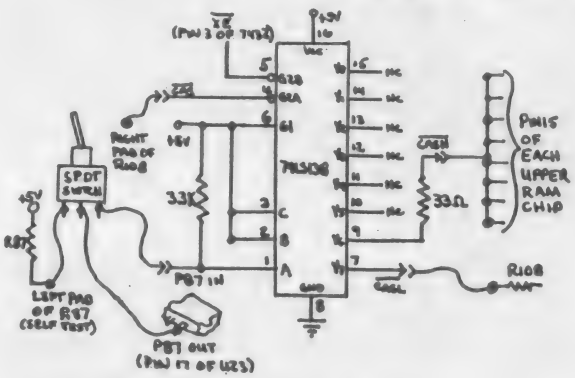
I understand a 512k RAM modification prototype has already been built. Now a half a mega-byte of memory to play with....who says the ATARI computer is nothing but a game machine...

BUILDING THE TERMINATOR-XL

By Mike Redmond

Paul has gone over the basics about how to build the 256K RAMBO XL. The 512K TERMINATOR XL is a bit more involved and might be considered a project best suited to hardware hackers. The same board used for RAMBO is used for TERMINATOR, but another chip and a few more resistors are added, along with another stack of 256K RAM. The (revised) TERMINATOR XL circuit is included in this ish.

Build up the complete board (with low profile sockets), including the optional 74LS138, 33 OHM and 3.3K resistors. I recommend using a chunk of light gauge, 6 conductor, stranded flat cable to connect all the PB signals. Bend out PB2 through PB7 on chip U23 (pins 12 through 17) and break off half a chip socket that can accomodate the 6 pins. Solder the first 5 socket pins to the PB2 through PB6 board connections. The 6th pin of the socket must go to the center position of a two way (SPDT) switch (this can be mounted near the left rear of the cabinet bottom). One of the ends of the switch must be wired to the PB7



MODIFIED TERMINATOR-XL CIRCUIT

To build the TERMINATOR you will need the RAMBO parts (74LS32, 74LS153, 74LS158 (can get from U27 socket), 2 capacitors and a 33 OHM resistor), 16 RAM chips (41256-15 or equiv), a 74LS138, 2 more resistors (33 OHM and 3.3K) and a miniature SPDT toggle switch. I also strongly recommended a set of 8 IC chip heat sinks, some flat ribbon stranded cable (6 conductor) and some fine gauge wire wrap wire.

You will probably want to start by stacking your RAMS. You will have to devise a heat sink to go between the chips (there are chip heat sinks that can serve this purpose). We found a non-sinked version wouldn't stay happy for more than about 6 hours. For each of the 8 256K chip sets, slip one over the other (with heat sink in between) and solder the upper chip's pins to the lower chips pins (1 to 1, 2 to 2, etc.). But DON'T solder pin 15 of the upper chip to the lower chip. That pin does the bank select function and must be bent out from the stack. Set the stacks aside for the moment.

lead out of the board. The other end of the switch must have a lead soldered to the left pad of R67 (where PB7 from the PIA used to go). Install the socket connector on the U23 pins.

Attach lead wires to the remaining connection pads on the board and install the board into it's socket (U27). Remove resistor R32 and solder one end of a 33 OHM resistor into the right R32 pad. Solder the RAS lead from the board to the "dangling" end of this resistor. Unsolder and lift the right side of R108. Solder the CAS lead from the board to the right R108 pad and solder the CASL lead to the dangling end of R108 (this is the lower bank select signal).

Now for the RAM stacks. Install the first stack in the U13 socket. Try to finaly attach (without soldering) the CASH lead from the board onto the bent out pin 15 on the upper chip. Also finaly attach a piece of light gauge, solid, insulated, easily strippable wire (I used wire-wrap wire)

to the same pin (this will be the upper CAS bus...make it long enough to reach the pin 15 of all 8 RAM stacks). Solder the 2 wires to pin 15. Install the next stack in U14 and solder the BUS wire to pin 15 of this stack. Proceed in this way until you've installed the last stack in U9. AND YOU'RE DONE!

Close up the box (if you can). You may be able to reinstall the shielding first if you are an electronics gymnast. It's tricky. To power up, you must have the TERMINATOR switch so the R07 lead is connected to PB7 from the PIA. Otherwise the screen will stare blankly back at you (while the self test goes nuts). As SOON as a screen with cursor appears, shift the switch into TERMINATOR mode and, OFF YOU GO.

Kurt Grittner has come up with DOS 2.5 patches that generate 4 RAMDISKS on the TERMINATOR (D5: through D8:). All are 707 sectors long. QUITE THE SYSTEM FOR A BBS OR ANIMATION SYSTEM!

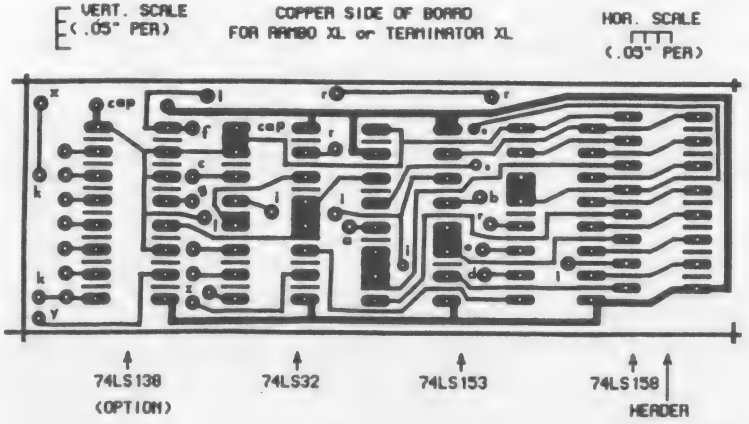
And if that's not enough, get out your fans and heavy duty power supplies. The same technique can be used to stack up up to 8 sets of 256K chips (THAT'S TWO MEGA BYTES). To do this, chop the tie high paths from pins 2 and 3 of the 74LS138 and tie those select inputs to PA7 and PA6 of U23 (pins 9 and 8). Then pins 10 through 15 of the 74LS136 can be used to select (through CAS pin 15 on the RAMS of each bank) 6 additional 256K banks. WARNING...SEVERE RAM MELTDOWN POSSIBLE! And, for software, you're pretty much on your own.

ENJOY (TO THE MEGA MAX)...MJR

COMMUNICATIONS CORNER

The following program although intimidating in appearance, is a pretty good (and short), modem program. It will work on almost any modem but you may be required to change the device name "R", depending upon the particular brand of modem you have. The program is one of those 'just lying around on a disk somewhere programs.

```
5 DIM IN$(100),IN2$(100)
9 XIO 38,#5,0,0,"R:"
10 XIO 36,#5,10,0,"R:"
15 OPEN #1,8,0,"E:"
20 OPEN #5,13,0,"R:"
30 XIO 40,#5,0,0,"R:"
40 PRINT #5;"ATE0Q0V2S7=255S11=50"
50 GOSUB 500
60 PRINT "PHONE NUMBER-"
70 INPUT IN$
80 PRINT #5;"ATTD";IN$;",,"
```



A--PB2	D--PB5	Z--$\overline{RA8}$	J->J = JUMPER
B--PB3	E--PB6	Y--$\overline{CASL}$ (OPTION)	K->K = 33 OHM RES. (OPTION)
C--PB4	F--PB7 (OPTION)	X--$\overline{CASH}$ (OPTION)	L->L = 3.3K RES. (OPTION)
	G--$\overline{CAS}$ (OPTION)		

RAMDISK SYSTEMS
by Levin Soule'

HAVE NOW 5

The number of cost free ramdisk systems available to users of 800XL's (and other Atari's) is now up to five. They are:

One - The "V" handler. See the June 85 ANALOG. It uses all unused user RAM in any Atari 400-130XE(main bank) as a super fast cassette recorder. You LIST or SAVE"V:", or ENTER or LOAD"V:", etc.

Two - The "X" handler. See the JULY 85 HAUG newsletter. Uses 14 of the 16K under the operating system as a fast cassette recorder, same as "V", but not as fast. Both "X" and "V" can be active at the same time.

Three - The BASIC8K.RAM program in the June 85 newsletter, with correction in the September newsletter. It provides 64 sectors of 128 bytes each, under the 8K of BASIC. The program must be part of your program. Any RAM sector can be saved to or read by addressing its RAM sector number (0-63).

Four - Drive number eight (D8:), a true RAMDISK of 128 sectors. This program and "X" can't be used at the same time. However, it or "X", plus "V" and BASIC8K.RAM can be used together in an XL. In an XE, you could use the 64K RAMDISK on the DOS 2.5 disk, plus "V", "X", and BASIC8K.RAM, all at the same time for upwards of 120K RAM (in theory). Do not expect super fast operation of "D8:" on the XL. LOAD and SAVE are fast. LIST is quick, but ENTER is almost as slow as a disk LOAD. Still it beats disk speed and saves your drive. The procedures to setup D8 are as follows. They have been in several newsletters and a magazine. Boot up in BASIC with Dos 2.5. POKE 1802,PEEK(1802)+128 to add D8 to the drive list. Press RESET and call DOS. Check the directory for D8. It should say 000 free sectors. Format D8 and check the directory again. It should say 499 sectors available. Write DOS to D8. Delete DOS.SYS from D8. Return to BASIC. POKE 5439,56. (Tells DOS to look to D8 for DUP.SYS. I got total lockup once by not making this POKE.) Call DOS. Write new DOS files to your disk. Turn off the computer. Boot up in basic. Go to DOS. Check the directory of D8. It should show 000 free sectors. It will have to be formatted before using. You can do this from DOS or from an early line in a program (or direct from BASIC) by entering XIO 254,#1,0,0,"D8:". If you check the directory of D8 it will show 499 available sectors, but there are only 128 available. If you try to use more then 128, the computer will crash, but good! You can write DOS files to D8, then delete DOS.SYS, and create MEM.SAV in D8 by first going to BASIC and POKE 5439,56. Then go to DOS and create MEM.SAV. From then on you will have instant DUP.SYS and MEM.SAV. Same as in the up coming number five. A point of interest: Location 5439 tells DOS which drive to access for DUP.SYS and MEM.SAV. A 49 here is drive one, 50 is drive two, and 56 is drive eight. When program developing you could POKE 5439,50 and use drive two as the utility drive, therefore not using any space on your program disk and also still be able to use both "X:" and "V:".

DRIVE MAP

BIT	7	6	5	4	3	2	1	0
DRIVE OFF	0	0	0	0	0	0	0	0
DRIVE ON	1	1	1	1	1	1	1	1
DRIVE NR	8	7	6	5	4	3	2	1

PEEK(1802)=3 for bits 0 and 1 on. Therefore drives 1 and 2 are active.

PEEK(1802)=131 for bits 0, 1, and 7 on. Therefore drives 1, 2, and 8 are active.

PEEK(1802)=15 for bits 0, 1, 2, and 3 on. Therefore drives 1, 2, 3, and 4 are active.

PEEK(1802)=255 for all bits on. Therefore drives 1 thruogh 8 are all active.

Five - DOS 2.5XL, modfies DOS 2.5 so it loads DUP.SYS and creates a MEMSAVE file under the operating system. When you boot the disk, DUP.SYS is loaded and you can go back and forth between DOS and your program in a flash.

DOS 2.5XL
Feedback, Aug 85, Adelaide, Australia
(ACE of Eugen, Oct 85)

When used with the XE computers, DOS 2.5 can make use of the extra memory available as a ramdisk. In addition, both DUP.SYS and MEM.SAV are stored on this ramdisk which allows instat access

to DOS and automatic saving of any program in memory without a (normal) disk access.

This latter feature can be imlemented on XL machines with 64K of memory by using the RAM behind the operating system ROMs. A program to do this with DOS 2.0s was published in number 24 of Analog magazine. The program presented here makes the same modification to DOS 2.5.

When RUN, the program will create a file PATCH25.OBJ. Boot with DOS 2.5 and using option L, load the file PATCH25.OBJ. Use option H to save your patched version of DOS 2.5. Reboot using this disk, and you can now go between DOS and a EASIC program in a flash.

```
0 REM SAVE"D:SWITCH2.5"
10 REM Modification to DOS 2.5 to
11 REM store DUP.SYS and MEM.SAV
12 REM in the bank switch RAM
13 REM behind the OS ROM from $CC0
14 REM to $F8FF
16 REM This mod for 64K XL's only
20 REM Adapted from ANALOG #24 by
21 REM Robert Luce
23 REM ********************************
24 REM written by Alec Denson 6/85
30 REM from FEEDBACK ADELAIDE Atari
31 REM Club, Box 333, Norwood,
33 REM Australia S.A. 5067 Aug '85
34 REM ********************************
40 REM REPRINTED ACE Newsletter
41 REM 3662 Vine Maple, Eugene, OR
42 REM and HAUG Newsletter
43 REM 3911 W. Crestview, Huntsville,
44 REM AL.  35816
100 CK=0:DIM A$(339)
105 ? :? "Reading Data...."
110 FOR I=1 TO 339
120 READ A
130 CK=CK+A
140 A$(LEN(A$)+1)=CHR$(A)
150 NEXT I
160 IF CK<>41072 THEN ? "ERROR IN DATA
STATEMENTS-CHECK TYPING":END
170 OPEN #1,8,0,"D:PATCH25.OBJ"
:PRINT #1;A$:CLOSE #1
1000 DATA 255,255,231,20,233,20,32,192
1010 DATA 23,70,23,130,23,32,05,24
1020 DATA 169,C,133,212,133,214,169,29
1030 DATA 133,215,169,192,133,213,162,16
1040 DATA 32,119,24,169,216,133,213,162
1050 DATA 7,32,119,24,32,70,24,96
1060 DATA 169,0,133,212,169,224,133,213
1070 DATA 150,0,162,3,177,212,72,32
1080 DATA 05,24,104,145,212,32,70,24
1090 DATA 200,208,241,230,213,202,16,236
1100 DATA 96,234,182,23,C,24,240,73
1110 DATA 32,70,23,200,158,23,48,65
1120 DATA 32,108,21,32,105,23,169,255
1130 DATA 141,158,21,141,157,21,162,16
1140 DATA 169,47,157,68,3,169,24,157
1150 DATA 69,3,32,164,21,32,85,24
1160 DATA 162,21,169,0,133,212,133,214
1170 DATA 169,31,133,215,169,228,133,213
1180 DATA 32,119,24,32,70,24,169,0
1190 DATA 141,157,21,141,158,21,70,146
1200 DATA 25,19,24,39,24,32,05,24
1210 DATA 169,0,133,214,133,212,169,228
1220 DATA 133,215,169,31,133,213,162,21
1230 DATA 208,18,58,24,140,24,32,119
1240 DATA 24,32,70,24,206,157,21,76
1250 DATA 152,32,32,162,24,88,169,112
1260 DATA 141,14,212,169,10,141,14,210
1270 DATA 96,120,169,0,141,14,212,141
1280 DATA 14,210,173,1,211,41,254,76
1290 DATA 107,24,173,1,211,9,1,141
1300 DATA 1,211,56,234,234,234,234,32
1310 DATA 156,25,90,160,0,177,214,145
1320 DATA 212,200,208,249,230,213,230,215
1330 DATA 202,208,242,56,234,234,234,234
1340 DATA 234,234,234,234,234,234,234,63
1350 DATA 25,108,25,32,65,24,169,0
1360 DATA 133,212,133,214,169,29,133,213
1370 DATA 169,192,133,215,162,16,32,119
1380 DATA 24,169,216,133,215,162,7,32
1390 DATA 119,24,32,70,24,96,234,234
1400 DATA 234,234,234,234,234,234,234,234
1410 DATA 234,234,49,31,53,31,178,174
1420 DATA 181,216,204
```

As in the past, a working copy will be on the club BBS.

DYTE MAGAZINE EDITOR IN CHIEF VISITS ATARI AND SPEAKS ABOUT THE 520ST

West LA Atari UG, Oct. 85

Byte Magazine will report on the ST by the end of the year. We don't have any quotes from that one yet, but judging from the fights their editors have had over who gets to play with it next, it should be a goodie. Byte's Editor-in-chief, Phil Lemmons, visited Atari's engineering and software departments in August and had this to say afterward:

"I visited Atari yesterday afternoon and got my first really good look at an ST520. I'm extremely impressed.

ACEEO NOV 85

SECTOR

The Missing Link

reprint Page 6t

Ron Lew

The idea of SECTOR is to allow those of you with an Atari 810 disk drive to experiment without being limited by DOS to the file structure. With SECTOR you are able to load, edit, and save ANY sector on the disk. With a reasonable understanding of DOS II's file structure you can perform all kinds of "nifty" things, such as retrieving deleted files and repairing damaged files. Examining and altering auto-boot disks is also greatly simplified. As an understanding of the way data is stored by DOS II will be of help, I will briefly outline its file structure.

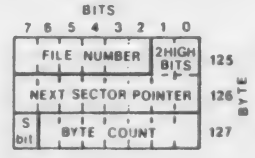
The 810 drive organizes the floppy disk as a collection of numbered blocks of bytes called "sectors". There are 720 sectors or blocks, and each holds 128 bytes or characters. As each file is created, an empty block is found and the data is poured into it. When the sector is filled, for another free sector has to be allocated and somehow linked to the first so when the file is being read the second sector can be found.

The directory information for this file only tells the DOS where to find its first sector. So how does DOS find the rest of the file? Well, only 125 bytes in each block are used for the user's data. The remaining 3 bytes are kept and used by DOS to provide 3 functions: 1. To point to the next sector in the file, 2. To say which number file the sector belongs to, and 3. To indicate if the sector is a "short" sector, and if so how many bytes are valid.

The pointer is obviously the key to the way in which DOS finds the next sector allocated to the file. The second function is not really essential, but it is useful, because as DOS created the file it notes the occurrence of the file's name entry in the directory and places this number into one of the last 3 bytes of each sector used by the file. Whenever the file is read back, if there is ever a discrepancy between the value of this byte and the directory, DOS assumes there has been some problem. It will report this to the user as the dreaded "ERROR 164", file number mismatch.

This unhappy event is usually caused by the careless user either "BREAK"ing or "SYSTEM RESET"ing during a disk operation, or swapping disks in a drive while a file is still open on the drive. Both are "NO NO" mistakes which should be avoided at all costs!

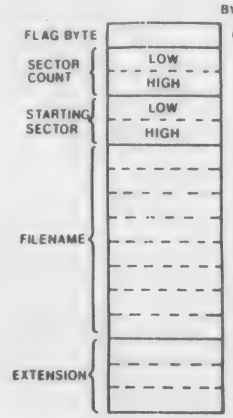
The last function is a vital one, for a file may not have used all the bytes in its last sector, and if this is the case DOS needs to know this fact and how many bytes of that sector are allocated to the file. There are 3 functions and 3 bytes, so it seems logical to have one byte per function. This cannot be so, however, because there are 720 sectors on a disk. So more than one byte is needed to store the next sector information. Since the directory position byte number does not have to be larger than 63, it does not require all 8 bits of its byte, so two of its bits are used by the next sector pointer.



This is how the sector's bytes are allocated. Looking at the diagram you will see one of the bits has not been explained yet: the "S" bit in byte 127. If the last sub-total of a file is not completely used, then the S bit is set to logic high, and the BYTE COUNT will give the actual number of valid bytes.

THE DIRECTORY

There are 8 sectors (361-368) allocated to the disk directory, each holding 8 entries, or 64 entries total. The 16 bytes of each file directory are allocated as follows:



The flag byte is used to indicate the status of the file and the bits are mapped as follows:

BIT No * IF SET HIGH, THEN
BIT 7 * FILE HAS BEEN DLLETED
BIT 6 * FILE ENTRY EXISTS
BIT 5 * FILE IS LOCKED
BIT 0 * FILE IS OPEN FOR OUTPUT

The flag byte is used to indicate the status of the file and the bits are mapped as follows:

BIT # ... If set HIGH, then
7 File has been deleted
6 File entry exists
5 File is locked
0 File is open for output

Thus the flag byte may have the following values:

VALUE	STATUS
$00	Entry not yet used
$40	Entry in use (normal closed file)
$41	Entry is in use (and file is currently open for output)
$60	Entry is in use (AND file is locked)
$80	Entry is available (prior file has been deleted)

The sector count (number of sectors in the file) and the starting sector number are obvious, as is the filename. Note, however, that DOS does not insert the full stop before the extension. The directory manager routines remove and insert this for the user's convenience.

THE PROGRAM

The precise format of the program is important, so be careful to include the correct number of spaces and characters where applicable, otherwise you may find some strange numbers will result. The program, when first RUN, will take over 10 seconds to initialize its string and arrays, so if you BREAK out of the program you can resume it by typing GOTO 100 and avoid the long 10 second wait. This continues without having to re-initialize! The program is based around a menu and has 5 options:

1 Normal Directory Listing. This gives the standard disk directory listing in two column format. Typing RETURN will take you back to the menu. Any other key will re-run the directory.

2 Load Sector. This allows you to load a sector into the buffer, and will first ask you for the sector number in HEX, i.e. 001 to 2D0. Typing RETURN only will cause the program to ask for a decimal value in the range 1 to 720. A further RETURN will allow you the option of loading the current sector by typing ...

3 Save Sector: This is identical to the load sector option in use

4 Edit Sector: This is the major section of the program. The current buffer contents are displayed in the form of a matrix, and there are several options available. These include:

Pressing START aborts the matrix display and asks for the X and Y coordinates to edit;

Pressing OPTION suppresses the printing of the hexadecimal buffer listing, but still gives the character;

Pressing SELECT suppresses the printing of the character buffer listing, but still gives the hexadecimal byte value table.

When the program asks for the X co-ordinate to edit, the following commands are available:

+ loads and displays the next disk sector

- loads and displays the previous disk sector.

N loads and displays the next sector in the same file as the current sector (if valid).

P dumps the display to a printer.

If none of these options are required, you can either type RETURN to get back to the main menu or type the X co-ordinate of the byte you wish to alter. You will then be asked for the Y co-ordinate, after which you can insert the hex or decimal number, or an ASCII string.

5 Examine Directory Sectors: This allows you to examine the disk's directory sectors directly. It prints out the flag byte, the number of sectors, and the starting sector for each file entry to the screen. If the START key is held down while this is entered, everything is printed to the printer rather than to the screen. Pressing START after the routine has been entered will pause the output to screen or printer. SELECT will restart at the first directory sector, and OPTION will return you to the main menu.

USING SECTOR

Quite apart from simply experimenting and learning about the disk system, there are many practical uses for SECTOR. For instance, if you have accidentally deleted an irreplaceable file on a disk, it can be retrieved by finding its old directory allocation using option 5 (examine directory sectors), then using the sector edit facility to alter the FLAG byte to 940. You should then copy off to a fresh, formatted disk all the desired files.

One important part of the ATARI DISK SYSTEM is the VOLUME TABLE OF CONTENTS (sector 360) in which DOS keeps track of which sectors are in use and which are free for new or extended files. This is the subject of another article. Meanwhile I hope you have many interesting hours of experimentation.

– Ron Levy

Solder two wires to the two conductors on the jack. Join the two cut wires from the speaker with the ones from the jack. Make sure you have got the right wire. Swig where. In other words, the connection to the speaker is the same except in the middle the leads of the jack are attached to pick up the signal. You've got to protect these, so either wrap electrical tape around them or use those insulated connectors.

Now you're done with the TV set. It's time to solder two wires to the two conductors of the plug. Hey! You with the earphone in your TV, wake up! You've got to do this too! Ok, now you have two wires coming from your plug. Take these two wires and wrap them to the two wires on the side of the transformer which has two wires (The other side has 3, that's the 1000-ohm side). Once again, protect these connections.

Take the other side of the transformer, the one with 3 wires, and connect the two on the outside to the red and green wires of the phone line. You can do this anyway you want, I leave it up to you. I suggest protecting the transformer somehow. Mine is in my phone. Well, you're done!

TONE DIAL is a program which generates the tones and puts them through the speaker. However, I didn't write this. It's from (*Gasp*, Copyright!) A.N.A.L.O.G. Magazine issues 19 and 21.

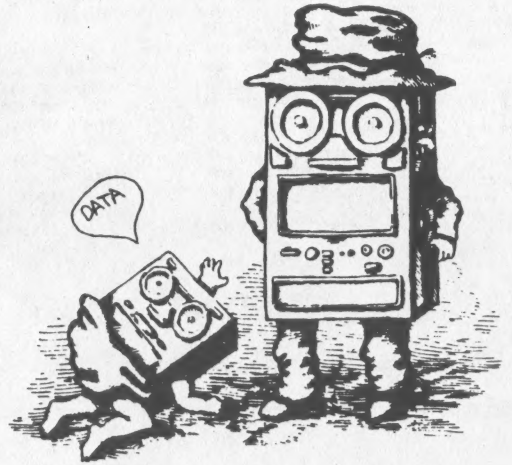
Theoretically, this should work with ANY modem if you can put the subroutine into the terminal program. I know it works on the 835. I own one! So, now you can tone dial like the big modems do.

FSIZE BY RALPH MALDEN

```
/* FSIZE.C - returns size of file */

#include "getname.c"

main() {
  char name[20];
  int iocb,len,divisor,kbytes,remain;
  fast();
  iocb=getread(name," ");
  divisor= (peek(0x1311+name[1]-'0'))
```




```
ensity sectors *

,highmem()-

number of K */
0)/1024;
es, %d.%dK\n",
);
 %d bytes\n\n",
r);

 inverse

=128;
: */
```


SECTOR

```
0 REM ##############################
1 REM .  A Disk Utility Programme.
2 REM .    Copyright MAPSOFT Ltd.
4 REM .          By Ron Levy.
5 REM Reprinted from the U.K Atari
6 REM Newsletter, Sussex, England
7 REM by the [illegible]
8 REM [illegible]
      ##############################
9 BUFF=1536:UNIT=1:POKE 709,15
10 DIM F$(38),OPT$(10),HEX$(512),T$(16
),HVAL$(38),H$(4),Y$(4),SECTOR$(10),DI
SK$(5),B(7),BYT$(256)
12 FOR N=1 TO 5:READ Y:DISK$(N,N)=CHR$
(Y):NEXT N
18 ? "STAGE 1"
19 REM ((( SET UP HEX CONVERTER )))
20 T$="0123456789ABCDEF"
22 FOR N=1 TO 256
24 Y=INT((N-1)/16):Y2=N-Y*16:Y=Y+1
26 L=LEN(HEX$)+1:HEX$(L,L)=T$(Y,Y)
27 L=LEN(HEX$)+1:HEX$(L,L)=T$(Y2,Y2)
28 NEXT N
30 C=53279:REM Consol Switches.
35 ? "STAGE 2"
40 REM (( Set Up Character Array ))
41 FOR BYT=1 TO 255:BYT$(BYT,BYT)=CHR$
(BYT)
44 IF BYT)26 AND BYT(32 OR BYT)124 AND
 BYT(128 OR BYT)154 AND BYT(160 OR BYT
)252 THEN BYT$(BYT,BYT)=CHR$(0)
46 NEXT BYT
50 REM (((( Create Display List ))))
52 D=561:D0=PEEK(D):D1=D0-1:DL=PEEK(56
0)+D0*256:DL1=DL-256
53 FOR A=1 TO 6:POKE DL1+A,PEEK(DL+A):
NEXT A
54 FOR A=6 TO 50 STEP 2:POKE DL1+A,0:P
OKE DL1+1+A,2:NEXT A
55 FOR A=52 TO 53:POKE DL1+A,PEEK(DL+A
-23):NEXT A
56 POKE DL1+54,PEEK(561)-1
100 REM (((((((((( Main Menu. ))))))))
102 GRAPHICS 0:POKE 709,15:CHD=0
105 ? "          Sector Utility."
106 ? "          ================";:?
108 ? "            By Ron Levy.";:?
112 ? "    Disk Directory ..... (1)"
114 ? "    Load Sector ........ (2)"
116 ? "    Save Sector ........ (3)"
118 ? "    Edit Sector ........ (4)"
120 ? "    Examine Directory ... (5)"
150 POSITION 12,20:? "Option --)";
160 CLOSE #2:OPEN #2,4,0,"K:":GET #2,K
:CLOSE #2
170 X=K-48:IF X(1 OR X)5 THEN 100
180 ON X GOTO 1000,2000,3000,4000,5000
200 FOR I=1 TO SIZE
1000 REM ((((((( DISK DIRECTORY ))))))
1020 GRAPHICS 0:POKE 709,15
1100 TRAP 1600:OPEN #1,6,0,"D:*.*"
1110 INPUT #1,F$:? F$;"  ";
1120 INPUT #1,F$:? F$:GOTO 1110
1600 CLOSE #1:INPUT #5:IF X$="" THEN 1
00
1630 GOTO 1000
2000 REM (((((((( Load Sector )))))))
2100 GRAPHICS 0:POKE 709,15:PRINT :TRA
P 40000
2110 ? "       LOAD Sector Routine.";:?
2200 ? " Which Sector (HEX) --) ";
2210 INPUT SECTOR$:IF SECTOR$="" THEN
PRINT :GOTO 2400
2220 IF LEN(SECTOR$)()1 THEN 2000
2230 I=ASC(SECTOR$(1,1)):J=ASC(SECTOR$
(2,2)):K=ASC(SECTOR$(2,2))
2240 IF I(65 THEN I=I-48:GOTO 2260
2250 I=I-55
2260 IF J(65 THEN J=J-48:GOTO 2280
2270 J=J-55
2280 IF K(65 THEN K=K-48:GOTO 2300
2290 K=K-55
2300 SECT=I*256+J*16+K
2310 IF SECT(1 OR SECT)5720 THEN ? CHR
$(253):GOTO 2000
2330 SECTOR=SECT:GOSUB 10000:GOTO 100
2400 ? " Which Sector (DEC) --) ";
2410 INPUT SECTOR$:IF SECTOR$="" THEN
PRINT :GOTO 2500
2430 TRAP 2400:SECT=VAL(SECTOR$):TRAP
40000
2440 IF SECT(1 OR SECT)720 THEN ? CHR$
(253);:GOTO 2400
2450 SECTOR=SECT:GOSUB 10000:GOTO 100
2500 ? "TYPE N TO LOAD SECTOR ";SECTOR
;" ";
2520 INPUT SECTOR$:IF SECTOR$()"N" THE
N 100
2530 ? :? "Ok. -- Loading Bou....";
2550 GOSUB 10000:GOTO 100
3000 REM (((((((( Save Sector )))))))
3100 TRAP 40000:CHD=1:GRAPHICS 0:POKE
709,15:PRINT
3110 ? "       SAVE Sector Routine.";:?
3200 ? " Which Sector (HEX) --) ";
3210 INPUT SECTOR$:IF SECTOR$="" THEN
PRINT :GOTO 3400
3220 IF LEN(SECTOR$)()1 THEN 3000
3230 I=ASC(SECTOR$(1,1)):J=ASC(SECTOR$
(2,2)):K=ASC(SECTOR$(2,2))
3240 IF I(65 THEN I=I-48:GOTO 3260
3250 I=I-55
3260 IF J(65 THEN J=J-48:GOTO 3280
3270 J=J-55
3280 IF K(65 THEN K=K-48:GOTO 3300
3290 K=K-55
3300 SECT=I*256+J*16+K
3310 IF SECT(1 OR SECT)5720 THEN ? CHR
$(253):GOTO 3000
3320 SECTOR=SECT:GOSUB 10000:GOTO 100
3340 GOTO 100
3400 ? " Which Sector (DEC) --) ";
3410 INPUT SECTOR$:IF SECTOR$="" THEN
3500
3430 TRAP 3400:SECT=VAL(SECTOR$):TRAP
40000
3440 IF SECT(1 OR SECT)720 THEN ? CHR$
(253);:GOTO 3400
3450 SECTOR=SECT:GOSUB 10000:GOTO 100
3500 PRINT
3510 ? :? "Type N To SAVE Sector ";SEC
TOR;" ";
3520 INPUT SECTOR$:IF SECTOR$()"N" THE
N 100
3530 ? :? "Ok. -- SAVEing Bou....";:GO
SUB 10000:GOTO 100
4000 REM (((((((( EDIT SECTOR )))))))
4040 GRAPHICS 0:POKE 709,15:POKE 8,81:
TRAP 40000
4050 ? "Sector)";:PRINT SECTOR
4055 POSITION 14,0:? "File)";
4060 BYTE=PEEK(BUFF+125):GOSUB 11000
4065 FILE=(BYTE-B(1)*2-B(0))/4:? FILE;
4070 NXTSEC=PEEK(BUFF+126)+256*(B(1)*2
+B(0))
4074 IF PEEK(BUFF+127))127 THEN NXTSEC
=0
4075 POSITION 23,0:? "Next Sec.)";NXTS
EC
4080 POSITION 0,1:? "$";
4085 H=INT(SECTOR/256):? HEX$(H*2+2,H
2+2);:H=SECTOR-H*256:? HEX$(H*2+1,H*2+
2);
4090 POSITION 10,1:? "$";HEX$(FILE*2+1
,FILE*2+2);
4100 POSITION 32,1:? "$";:X=B(1)*2+B(0
)
4105 ? HEX$(X*2+2,X*2+2);
4110 Y=PEEK(BUFF+126):? HEX$(Y*2+1,Y*2
+2)
4160 ? "  0  1  2  3  4  5  6  7  0123
4567";
4170 POSITION 3,3:? CHR$(17);
4180 FOR N=1 TO 8
4184 ? CHR$(18);CHR$(10);CHR$(32);
4186 NEXT N:? "--------";
4200 FOR L=0 TO 15 STEP 2
4210 POSITION 1,L+4:? L;
4220 POSITION 3,L+4:? CHR$(124);
4240 IF PEEK(C)=5 THEN 4389
```

```
4249 POSITION 4,L+4
4250 FOR PK=0 TO 7
4270 BYT=PEEK(BUFF+PK+L*8)
4280 ? HEX$(BYT*2+1,BYT*2+2);" ";
4290 NEXT PK
4300 REM ... PRINT CHARACTERS ...
4305 IF PEEK(C)=3 THEN 4400
4309 POSITION 20,L+4
4310 FOR CH=0 TO 7
4320 BYT=PEEK(BUFF+CH+L*8)
4330 IF BYT=0 THEN ? CHR$(0);:GOTO 439
0
4350 PRINT BYT$(BYT,BYT);
4390 NEXT CH
4395 IF PEEK(C)=6 THEN 4500
4400 NEXT L
4500 REM ...EDIT-IT...
```

```
4000 POSITION 4,22:? "
";
4005 POSITION 4,22:? "New (STRING) ";
4010 INPUT NVAL$:IF NVAL$="" THEN 100
4020 FOR X=1 TO LEN(NVAL$)
4030 POKE BUFF+EDBYT+X-1,ASC(NVAL$(X,X
))
4050 NEXT X:GOTO 4000
5000 REM <<<< Examine Directory >>>>
5020 GRAPHICS 0:POKE 709,15:TRAP 40000
:CNT=0:CLOSE #1
5030 IF PEEK(C)=7 THEN OPEN #1,6,0,"E:
":POKE 709,15:P=0:POKE 0,D1:GOTO 5100
5040 TRAP 100:OPEN #1,6,0,"P:":TRAP 40
000:P=1
5050 IF PEEK(C)<>7 THEN 5050
5060 P=1
```

```
5650 NEXT J:NEXT I
5000 ? #1;"     Re-run (Y) -->";
5010 INPUT N$:IF N$="Y" THEN 5000
5900 CLOSE #1:GOTO 100
5949 REM << A Little Delay! >>
5950 IF PEEK(C)=6 THEN 5950
5955 IF PEEK(C)=3 THEN POP :CLOSE #1:G
OTO 100
5960 IF PEEK(C)=6 THEN RETURN
5961 IF PEEK(C)=5 THEN POP :GOTO 5100
5965 IF PEEK(C)=7 THEN 5955
10000 REM <<<<< Disk Interface >>>>>
10100 POKE 769,UNIT
10110 IF CMD=0 THEN POKE 770,82
10120 IF CMD=1 THEN POKE 770,87
10130 POKE 772,BUFF-INT(BUFF/256)*256
```


(46)

NOV85

Simple Calculator

by Ed Smith & Dick Basso

It can be used as a Subroutine <-----> It can be used as a Subroutine

This short program is useful to those who wish to avoid the print command. The routine works from **Left to Right.**

This routine supports addition (+), subtraction (-), division (/), multiplication (*), and exponentiation (^). It does not support the use of Parentheses. Routines could be added to handle levels of Parentheses but then the program would no longer be a simple program.

```
5 REM CALCULATOR MODE BY ED SMITH
10 DIM P$(140):CALC=20:POKE 710,0:POKE
 709,0:GOTO 60
20 FOR I=1 TO L:A=ASC(P$(I))
22 IF A=43 THEN C=C+VAL(P$(I+1))
24 IF A=45 THEN C=C-VAL(P$(I+1))
26 IF A=42 THEN C=C*VAL(P$(I+1))
28 IF A=47 THEN C=C/VAL(P$(I+1))
30 IF A=94 THEN C=C^VAL(P$(I+1))
32 NEXT I:RETURN
50 TRAP 40000:? CHR$(125):POKE 712,52:
? CHR$(253):C=0:POSITION 8,4:? "    WH
AT YOU TYPED    ":? :? :? P$:?
52 ? :? "BE CAREFUL WHEN TYPING":? :?
"USE OF ( ) AND/OR [ ]":? "ARE NOT SUP
PORTED"
53 POSITION 6,20:? " PRESS ANY KEY TO
CONTINUE";
54 IF PEEK(764)=255 THEN 54
55 POKE 764,255
60 ? CHR$(125):? "      YOUR CALCULATIO
N AND ANSWER:     ":? :? P$;" = ";:? C:
? :? :POKE 712,198:? :?
61 ? "                 WORK AREA
   ":?
62 ? "ENTER CALCULATIONS":? "HIT RETUR
N WHEN DONE":? :INPUT P$:L=LEN(P$):IF
L=0 THEN END
65 TRAP 50:C=VAL(P$):GOSUB CALC:? C:GO
TO 60
```


Simple Calculator Continued

Program description:

Line 10: a long string P$ is DIMensioned to hold upto 140 characters of a problem (numbers and symbols). A variable CALC is set equal to 20. This is used in line 65 to GOSUB CALC, which means GOSUB 20.

Line 20: A FOR-NEXT loop is started with the FOR in line 20 and the NEXT in line 32. It will go through the loop as many times as the number of characters contained in P$. The length of P$ is determined in line 62 where L is set equal to the LENgth of P$. The variable A is then set equal to the ASCii value of the I-th character of P$ (I being the loop counter). For example if you are on the 5th loop the 5th character of the string is being examined and converted to its standard numeric ATASCII code.

Line 22: A=43 is the ATASCII numeric code for plus (addition). The IF statement tests for A=43 and if found to be 43 THEN C will equal the previous value of C PLUS the VALue of the next character in P$ (I+1).

Line 24: A=45 is the ATASCII numeric code for MINUS (subtraction). The IF statement tests for A=45 and if found to be 45 THEN C will equal the previous value of C MINUS the VALue of the next character in P$ (I+1).

Line 26: A=42 is the ATASCII numeric code for MULTIPLY. Same explanation as above except C will equal the previous value of C MULTIPLIED by the VALue of the next character in P$ (I+1).

Line 28: A=47 is the code for DIVIDE. Same explanation as above except will now divide.

Line 30: A=94 is the code for exponential. Same explanation as above except previous C will now be raised to the power VALue of the next character in P$.

Line 32: NEXT I sends the program back again to line 20 and continue to do so until it has counted up to L, the number of characters in P$. (see Line 20 above).

Line 50: TRAP 4000 resets the TRAP called for in Line 65. CHR$(125) clears the screen. POKE 712,52 sets the border color to red. CHR$(253) sound the bell. The information between the quotes are printed in inverse video (This includes the spaces).

(47)

Simple Calculator Continued

Line 53: POSITION 6,20 will print the line following at position 6 in the X direction and on line 20 in the Y direction. The information printed between the quotes is in inverse video.

Line 54: Address 764 contains the Key code value for the last key pressed. Used here to hold the screen display until a key is pressed before moving on. If no key is pressed then the program will continously loop on this line.

Line 55: Resets Address 764 to 255 so that the code for the key that was pressed is removed.

Line 60: ? CHR$(125) clears the screen. The information between the Quotes is typed in inverse video. Poke 712, 190 changes the screen border to a green color.

Line 61: The information typed between the quotes is in inverse video.

Line 62: P$ is the calculation that you want performed. The IF L=0 THEN END says that if you DO NOT enter a calculation and hit <RETURN> the program will end.

Line 65: TRAP 50: This is the trap statement that sends you to Line 50 if you enter something other then numbers and "legal" symbols (+, -, *, /, and ^). GOSUB CALC: Calc is a named variable that was set to 20 (see Line 10). This means to gosub to line 20 which is the calculation routine. The GOTO 60 just starts the whole process again for a new calculation.

Sample Problems

What is the effective interest rate if the basic rate is 10% compounded daily? Enter as follows:

.10/365+1^365-1*100

Answer: 10.515404

Enter the following:

1+5/2*4.55-3.22

Answer: 10.43

Remember it reads left to right one operation at a time.

Ed has incorporated the main routines in a home checking account program that he uses.

Title of Article

by Authors Name

Some thing that applies to the article or just Space Probes Space Probes

SPACE FILLER

(48)

Spreadsheets for the Mathematician

by Donald Forbes - JACG

oct 85

Ever think of using a spreadsheet package, such as Visicalc, to do numerical analysis? Or research mathematics? Or teaching math at the college or graduate level? Crazy??? Believe me, it works!

Albert Einstein said that "every physicist should have a shoemaker's job." So if your shoemaker's job is doing spreadsheets (today's equivalent of economics, which used to be called the 'dismal science') when you would rather be doing mathematical research, then rejoice! You have been liberated at last.

The idea is stupidly simple -- once someone points it out to you. (I am assuming that you know how a spreadsheet works.) Suppose you want to compute a Fibonacci series which goes 1,1,2,3,5,8,... where each number is the sum of its two predecessors.

All you have to do is set up your formulas like this:

Formulas

	A	B
1	FIB NOs	RATIO
2	1	
3	1	+A3/A2
4	+A2+A3	+A4/A3
5	+A3+A4	+A5/A4
6	+A4+A5	+A6/A5
	Replicate	Replicate
22	+A20+A21	+A22/A21

Then you get your output (including the convergent ratio) like this:

Output

	A	B
1	FIB NOs	RATIO
2	1	
3	1	1
4	2	2
5	3	1.5
6	5	1.666667
22	2584	1.618034

That's the whole idea! Nothing could be simpler. To compute factorials you use column A as a counter and in column B you multiply by the factorial you computed in the preceding row of column B.

Want to do linear regression? Put your string of x values in column A, your string of y values in column B, x squared in column C, y squared in column D, the x and y cross products in column E and the computed regression line (MX+B) in column G. Could anything be easier?

This is just the beginning. Here is a list of the goodies: Fibonacci numbers, factorials, bisection algorithm, fixed-point algorithm, limits, differentiation, Newton's method, numerical integration, Taylor polynomials, differential equations, polynomial interpolation, matrix multiplication, Jacobi and Gauss-Seidel algorithms, dominant eigenvalue, eigenvalues and diagonalization, continued fractions, Euclid's GCD algorithm, binomial theorem, synthetic division, contour graphs, modular arithmetic, Russian peasant multiplication, statistics (mean, correlation, regression, confidence intervals), probability (Bayes' rule), algebra word problems, trigonometry, compound interest, personal finance model, simplex linear programming, game theory, matrix powers, systems of linear equations, boundary-value problems.

Where is this treasure trove? In a new $17 165-page McGraw-Hill paperback called Mathematical Applications of Electronic Spreadsheets by professor Deane E. Arganbright of the Whitworth College department of math and computer science in Spokane WA. His phone number is 509/466-1000 x474 if you want to call him.

He points out that any "spreadsheet program calculates values for each of the expressions and displays the data and the calculated values in a spreadsheet format on the computer screen. By changing the entries in the matrix a user can modify the hypotheses or parameters of the model..."

"The electronic spreadsheet can also be used effectively and creatively in mathematics. Algorithms that are recursive, iterative or adaptable to a table format can often be implemented easily and naturally on a spreadsheet, allowing the user to change initial values, step sizes, and other parameters and see the result of the changes instantly. Moreover, many applied mathematics problems can also be set up, analyzed, and solved on a spreadsheet. The what-if features of the spreadsheet make it useful in mathematical modeling, the design and study of algorithms, problem solving, and the study of mathematics."

The book is written cookbook style: for each of the 36 topics there is an introduction, followed by SPREADSHEET CONSTRUCTION, then USER INTERACTION, then EXERCISES and MODIFICATIONS, and finally REFERENCES. He assumes you know the math and makes no effort to teach it, but does give you 65 textbook references with page numbers, and a dozen references to the workings of spreadsheets. His demos are done in Visicalc, but will work with Multiplan or Framework or any of the other popular packages.

Arganbright sums it all up as follows: "There are a number of reasons why the electronic spreadsheet is an especially good means of integrating the computer into the study of mathematics:

"Spreadsheet operation is natural and easy to learn... Spreadsheets for algorithms follow the same format and techniques commonly used for doing the work by hand... The spreadsheet format makes algorithms and manipulations concrete and easy to visualize... The what-if capabilities of a spreadsheet program allow a user to modify parameters easily and see the effects of changes instantly... Learning how to use a spreadsheet program provides experience and skills that are valued in the business community... Moreover the range of possible applications seems unending..."

"Of course, there is another fundamental reason for doing mathematics on a spreadsheet: It's fun!"

(49)

COMPUTER RELATED VISION PROBLEMS

NOV 85

by Wm Schneider - JACG

Excerpts taken from articles in
Newark Star Ledger September 26, 1985
Somerset Messenger Gazette February 14, 1985

The University of California's School of Optometry has opened the first eye clinic specializing in problems associated with the use of video display terminals (VDTs). The main objective is to deal with the VDT's adverse effect on eye health and user productivity.

The clinic is supported by donations from AT&T and Westinghouse Electric Corp. Their concerns are the growing number of complaints of eyestrain, double vision, headaches and fatigue. This results in higher error rates and reduced speed and efficiency.

Patients are examined and asked to describe their workstations. This includes physical positions relative to the VDT's such as desk height and distance from the screen. Special attention is given to lighting, the patients sensitivity to glare, problems with eye movement and coordination and ability to focus on the screen.

According to Dr. William Moskowitz, a behavorial optometrist at 2 Park Avenue, Somerville, N.J.; "Even under the best working conditions at least half of VDT operators have complaints about vision related symptoms. They suffer headaches, eyestrain, blurred or double vision and even permanent vision problems." He cites the National Academy of Sciences report, "Video Displays, Work and Vision". Released in late 1984 it confirmed that VDT'S do not cause diseases or pathologies of the eyes. This report does state however, that people who already have vision problems, some of them very subtle, are very likely to experience vision-related complaints when using a VDT. Direct symptoms of VDT related vision problems are burning, itching, watery, pulling or irritated sensations of the eye, headaches, momentary blurred or double vision, or difficulty seeing clearly at distances after prolonged VDT work.

Dr. Moskowitz had the following observation; "In a sense we have Neanderthal vision for computer age work. Human vision developed to assure survival, to spot game, enemies, or opportunities at great distances." This disparity between our distance vision preference and the need to do near work is the main source of vision problems. The effort it takes to do near vision work is significantly greater than the effort required for distance work. It involves very complex eye aiming and the ability to use both eyes together, smoothly and simultaneously.

VDT users may require a change in their regular eye lens prescription or lens design or special lenses just for VDT use. People who wear glasses to clear their distance vision may find their precription is actually causing problems when they use a VDT. For most people, low power, focus-relaxing lenses prescribed specifically for their visual capabilities and for their own computer workstation can help. For people with certain visual skills problems, visual therapy may be benefical to develop the skill and resilience needed for VDT work.

(50)

The following suggestions are intended for the workplace but can be applied to home usage as well.

1- The way the VDT workstation is arranged can have a big impact on vision complaints. Simple steps such as eliminating glare from the surroundings and reflections on the screen can help a lot. Adjust the brightness and contrast on the screen to obtain the clearest display possible.

2-Screens should be positioned so that workers can look at them as they would a typewriter. Place the screen so the operator can occasionally look up into a distant space instead of a nearby wall for visual relief breaks.

3- Choose software with dark characters on a light background. They are easier to discern.

4- When selecting a monitor, check for flicker and jitter. Also examine the text. Fuzzy-edged letters on a screen result in a constant effort by the eyes to clear up what wasn't clear to begin with.

By 1986 about 35 million Americans will work with VDTs as a daily part of their jobs. Although eye strain may appear to be an occupational hazard, many of us spend more time than we care to admit staring at a "tube" at home as well.

VOTERS IN ACTION!

Charles P. Lichtenwalner - JACG

OCT 85

(51)

In the April 1985 Scientific American "Computer Recreations" column an interesting simulation described as a voting game is a suggested programming recreation. A rectangular grid is populated with a random scattering of "voters of two political parties" (or colors, or symbols.) The program then picks one of the voters on the grid and one of their eight nearest neighbors--all picks at random. Then "the voter's political persuasion becomes that of his neighbor, regardless of earlier belief."

The attached listing is an implementation of this simulation written in ACTION!. Having only a green monitor I choose spaces and #'s to denote the two political parties. I decided to use Graphics 1 to give a square grid. As noted by A. K. Dewdney the appeal is watching large blocks of votes developing which migrate around as the two parties struggle for dominance. Eventually the struggle collapses and the screen fills with a single character (party.) If you believe in contrarian voting as my wife does, you might try modifying the program to an antivoting game where the selected voter adopts an opposite opinion to that of his neighbor.

In a follow-up article in the July Scientific American Mr. Dewdney mentions that some people had trouble noting the collapse of the two party system. He mentioned that his program takes the better part of a day to reach this state. It is not mentioned how large a grid he used, but I generally get a collapse within 2 to 5 minutes with the 20x20 grid of Graphics 1. However, ACTION! is fast. I get about 1000 voter changes per second. I put in a print statement to let me know whenever 65000 changes have taken place.

Another suggestion you may want to try is to set up a multi-party (more than two) world. To do this simply change the data in the FIGS array as noted in the comment. As currently configured up to eight parties can be simulated. Whatever your modification, the interesting part is watching the changing display.

To paraphrase the League, "Get out and VOTERS!"

```
;  VOTERS     C. P. LICHTENWALNER 9/11/85
;AFTER A SUGGESTION IN COMPUTER RECREATIONS
;COLUMN OF "SCIENTIFIC AMERICAN" APRIL 1985

BYTE SAVMSCL=$58, SAVMSCH=$59,TYPRTY
;CHANGE THE FIGS ARRAY TO VARY THE CHARACTER
;PRINTED ON THE SCREEN
;OR BIAS THE INITIAL VOTER DISTRIBUTION
;OR TO SET UP A MULTI-PARTY SYSTEM
BYTE ARRAY FIGS=[0 0 0 0 10 10 10 10], J
INT ARRAY NGHBRS=[65515 65516 65517 65535 1 19 20 21],
NGHBR, CELL
CARD SCREEN,I,II,SCRNLOC

PROC INITGR()
;SET GRAPHICS MODE 1 AND FIND START OF
;SCREEN MEMORY
   GRAPHICS(1)
   SCREEN=SAVMSCL+256*SAVMSCH
RETURN

PROC LDVTRS()
;LOAD THE SCREEN WITH A VALUE FROM FIGS
   INITGR()
   FOR I=0 TO 400
      DO
         J=FIGS(RAND(8))
         POKE(SCREEN+I,J)
      OD
RETURN

INT FUNC ADJNGHBR(INT CELL, NGHBR)
;ADJUST VALUE SO IT FALLS BETWEEN 0-399
;i.e. ACCOUNT FRO WRAPAROUND
      IF (CELL+NGHBR)<0 THEN RETURN (CELL+NGHBR+400)
            ELSEIF (CELL+NGHBR)>=400 THEN RETURN
(CELL+NGHBR-400)
            ELSE RETURN (CELL+NGHBR)
      FI

INT FUNC PICK()
;PICK A RANDOM NUMBER BETWEEN 0-399
      CARD RANDOM
      BYTE RND
      DO
      RANDOM=PEEK(53770)
      RANDOM=(RANDOM LSH 1)!PEEK(53770)
      UNTIL RANDOM<400
      OD
RETURN (RANDOM)

PROC VOTERS()
      BYTE KEY
      CARD SCNPC,SCNPN
      LDVTRS()
      FOR II=0 TO 100
      DO
            PRINTF("65000 # %U VOTER CHANGES%E",II)
      FOR I=0 TO 65000
      DO

;PICK A CELL AT RANDOM THEN A NEIGHBOR
;AT RANDOM AND CHANGE THE CELL TO MATCH
;ITS NEIGHBOR
            CELL=PICK()
            NGHBR=NGHBRS(RAND(8))
            SCRNLOC=ADJNGHBR(CELL,NGHBR)+SCREEN
            TYPRTY=PEEK(SCRNLOC)
            POKE(SCREEN+CELL,TYPRTY)
      OD
      OD
RETURN
```